REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE January 1997	3. REPORT TYPE AND DATES COVERED Technical Report, 7/1/96-12/31/96

4. TITLE AND SUBTITLE

ADAPTIVE DECISION MAKING AND COORDINATION IN VARIABLE STRUCTURE ORGANIZATIONS

5. FUNDING NUMBERS

N00014-93-1-0912

6. AUTHOR(S)

Alexander H. Levis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Center of Excellence in Command, Control, Communications and Intelligence
George Mason University
Fairfax, Virginia 22030

8. PERFORMING ORGANIZATION REPORT NUMBER

GMU/C3I-182-IR

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 N. Quincy Street
Arlington , VA 22217-5660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

19970313 057

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

Progress in research on coordination in distributed making organizations with variable structure is reported. Results on distributed process coordination in adaptive command and control teams and results in formulating the problem of modeling the decision making process using influence diagrams are presented. A new task on Measures of Performance and Effectiveness are described.

14. SUBJECT TERMS

Decision Making; Organization; Colored Petri Nets; Influence Diagrams

15. NUMBER OF PAGES

60

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

CENTER OF EXCELLENCE IN
COMMAND, CONTROL, COMMUNICATIONS AND INTELLIGENCE

GEORGE MASON UNIVERSITY
Fairfax, Virginia 22030

SEMI-ANNUAL TECHNICAL REPORT

for the period

1 July 1996 - 31 December 1996

for

ADAPTIVE DECISION MAKING AND COORDINATION IN VARIABLE STRUCTURE ORGANIZATIONS

Grant Number N00014-93-1-0912

Submitted to
Dr. W. S. Vaughan, Jr. (3 copies)
Office of Naval Research
Code 342
800 North Quincy Street
Arlington, Virginia 22217-5000

Copies to:
Director, Naval Research Laboratory
Administrative Grants Office, ONR
DefenseTechnical Information Center

Submitted by:

Alexander H. Levis
Principal Investigator

March 10, 1997

GMU/C3I-182-IR

TABLE OF CONTENTS

1. PROGRAM OBJECTIVES

The objective of this research, as described in the 1993 and 1996 proposals and the previous progress reports, is the investigation of several issues related to adaptive architectures for command and control and the coordination necessary not only for the functioning of an organization, but also for managing the adaptation. In particular, an organization is coordinated through direct and indirect means. The direct means includes the set of decision rules that the organization members use and the commands that they issue to each other. Indirect means include the dissemination of information within the organization; for example, organization members may share information or they may inform each other as to the actions they plan to take or decisions they have made. Coordination becomes a complex issue in adaptive organizations; organizations that adapt their architecture to changes in the tasks, changes in the environment, or changes in the available resources. Not only do the decision rules and the information architecture have to work for each fixed structure, but the designer has to deal with the problem, a metaproblem, of coordinating the variability. This becomes a particularly difficult problem in organizations that exhibit substantial complexity and redundancy in their information structure. The redundancy is necessary both for robustness and for flexibility and reconfigurability. In order to address these problems a set of tasks were defined in the 1993 and 1996 proposals; some additional tasks were defined in the modifications that took place in 1995; they are described in the next section. In addition, some basic work in algorithms and Colored Petri Nets needs to be done to develop tools and techniques for supporting the analysis and design.

2. STATEMENT OF WORK

The statement of work, as outlined in the proposals, is given below.

Task 1 (A): Consistency and Completeness in Distributed Decision Making

Develop a methodology for analyzing and correcting the set of decision rules used by an organization with distributed decision making. The methodology is to be based on the modeling of the set of decision rules in the form of a Colored Petri Net and on the analysis of the net using S-invariant and Occurrence graphs. The ability to verify and correct the set of decision rules has direct impact on the extent of coordination needed in an actual organization and the resulting communication load.

Task 2 (A): Variable Structures: Heuristic Rules in the Lattice Algorithm Constraints

Develop a methodology for considering additional constraints in the Lattice Algorithm. Such constraints include the degrees of redundancy and complexity at the different processing nodes (to be derived from the DFS algorithm of Andreadakis), the projected response time of the organization, ands some user-specified constraints on connections between decision making units. Develop a procedure for checking the validity of such constraints and incorporate them in the Lattice Algorithm. Generalize the approach to multilevel organizational structures and to variable structures, where variable structures are obtained by folding together different fixed structures. The real focus of the task is to introduce these additional constraints as a way of containing the dimensionality problem inherent in flexibility and reducing the coordination requirements.

Design a symbolic interface for the Lattice algorithm. The interface would have the capability of interpreting natural language inputs entered by the user and will include some symbolic processing. The system will generate the interconnections matrices used as input to the Lattice algorithm. The designer would then use the various tests described in the proposal (such as DFS algorithm) to check the validity of the interconnection constraints and to make required modifications.

Task 3 (A): Dynamic Task Allocation in Adaptive C2 Architectures

The objective of this task is the application of CAESAR II (Computer-Aided Evaluation of System Architectures) to several problems associated with the organizational design of flexible Command and Control Architectures for Joint Operations in modern littoral warfare. Specifically, the tools and techniques developed for the design of distributed tactical decision making organizations and for designing adaptive information structures for such organizations are embodied in the software suite in CAESAR II. When a well defined organizational task is mapped onto the humans and machines constituting the organization, several problems of inconsistency, redundancy, and ambiguity arise that degrade organizational performance. The results of Task 1 - theory and algorithms - will be applied to this problem. Furthermore, the existence of CAESAR II with the enhancements of Task 2 makes possible the support of model-driven experimentation.

Task 4 (A): Graphical Representation of C2 Decision Making and Supporting Inference

The objective of this task is to use the rapidly developing field of influence diagrams and Bayesian networks to build graphical representations of decision making and supporting inference (intelligence) processing for command and control organizations. In particular, the emphasis is on distributed command and control organizations that are involved in rapidly evolving tactical situations and are likely to reorganize so as to remain effective. As this is our first effort in this area, our focus for this task will be on the representaiton of decision making and inference processes within command and control. The representation issues that we will adress are distributed, concurrent, asynchronous, and interconnected command and control elements; the evolution of decision making tasks throughout a typicl tactical mission; the exchange of information between elements associated with the evolution of time within a specific mission phase; and the impact of overarching the environmental and threat uncertainties that inhibit the effective paritioning of the command and control elements. This activity will lead to critical new representation ideas in influence diagrams and Bayesian networks, enabling later research that addresses partitionaing algorithms of decision making and inference tasks for the purpose of effective allocation of such tasks to command and control elements.

Task 5 (B): Distributed Process Coordination in Adaptive Command and Control Teams

5.1 Develop further and refine the algorithm for the derivation of coordination rules that support alternative back-up strategies.

As part of a current research task, an algorithm has been outlined for the derivation of the coordination strategies of decision makers in adaptive command and control teams. This is accomplished by utilizing a Colored Petri Net representation of the decision making organization.

5.2 Develop an intuitive user interface for this algorithm and incorporate it in the suite of tools that constitute CAESAR II.

CAESAR II is a suite of tools that support the analysis, design, and evaluation of organizational architectures. The developed algorithm will be incorporated in CAESAR II and a graphical user interface will be developed for the user to interact with the algorithm and specify either a predefined back-up strategy or formulate new ones.

5.3 Apply the algorithm to several realistic examples using a variety of back-up strategies.

Examples drawn from the Adaptive Architectures for Command and Control (A2C2) initiative will be used to test the algorithm and analyze the results.

Task 6 (B): Behavioral and Performance Evaluation of Adaptive Architectures

6.1 Extend the System Effectiveness Analysis methodology to include adaptive architectures for Command and Control.

The System Effectiveness Analysis methodology has been developed and applied to fixed structure systems, including organizations with fixed structure. Basic research is needed to extend the methodology to adaptive architectures.

6.2 Incorporate in CAESAR II the analytical and graphical tools (using COTS to the maximum extent possible) necessary to apply the methodology to the assessment of adaptive organizations.

6.3 Develop the analytical and computational constructs necessary to assess the sensitivity of the effectiveness measure to the design parameters that determine the System Locus and the mission parameters that determine the Requirements Locus

Task 7 (B): Modeling Decision Making Activities for Adaptive C^2 Architectures

7.1 Extend graphical model language for specifying Command and Control planning tasks to address both the time sequencing of plans and concurrent battle execution and battle damage assessment (BDA).

This task extends early work on modeling decision making in a distributed, hierarchical organization by combining influence diagrams and Bayesian networks with Petri Nets. To date, we have decomposed the difficult decision making tasks solved by joint task forces to address the needed interaction amongst the decision making tasks at a given level and across levels of the organizational structure. We will address the complex decision making task models needed to define concurrent activities within each cell and show the effects on hierarchical, distributed decision making.

7.2 Define and analyze algorithms for assigning decision making functions to the C^2 entities of the organizational architecture.

This activity will develop measures of effectiveness for these assignments that are based upon decision making quality and timeliness. Communication issues will be treated as requirements for transmitting data elements amongst organizational entities, rather than as a measure of effectiveness.

7.3 Develop decision making structures that can be used to analyze (1) concurrent battle planning and planning for BDA, and (2) concurrent time-sequenced planning effectively.

These structures will build upon work already described as dynamic decision networks that incorporate the use of both influence diagrams and Bayesian networks.

Task 8 (A, B): Information Dissemination

Progress reports will be submitted in accordance with ONR requirements. The results of this research will appear in thesis reports and in technical papers to be presented at professional meetings and published in archival journals. In addition, oral presentations will be given periodically at review meetings organized by ONR

Tasks designated with the letter A are from the 1993 proposal; tasks designated B are from the 1996 one.

3. RESEARCH PLAN

The research plan describes the strategy for meeting the program objectives. Specifically the research plan is organized around a series of specific well-defined research tasks that are appropriate for theses at Master's and Ph.D. levels. Individual students are assigned to each task under the supervision of the principal investigator. Additional staff from the C3I Center are included in the project whenever there is a specific need for their expertise.

4. STATUS REPORT

The focus of Task 1 is the development of a methodology for analyzing and verifying the set of decision rules used by an organization with distributed decision making. The methodology is based on the modeling of the decision rules in the form of Colored Petri Net and on the analysis of the net using S-invariant properties and Occurrence graphs. The results obtained for the two analyses, when applied to a specific form of decision rules, have been presented in the Ph. D. thesis of A. Zaidi that was submitted as the third semi-annual report (dated January 1995). The key paper from this work has been accepted for publication in *Automatica* and and is scheduled to appear in the February 1997 issue. In addition, a new algorithm for designing organizations of interacting decision makers has been developed as an alternative to the Lattice Algorithm. This new algorithm is based on genetic algorithms; it was presented in the fifth semi-annual report (GMU/C3I-168-IR, dated January 1996) and has been submitted for publication.

Task 2 has continued to be a focus of activity during this year. The recoding and revision of CAESAR II, the Computer Aided Evaluation of System Architectures suite of software algorithms and tools, has continued and the capabilities of the system have expanded substantially. CAESAR II has been demonstrated to a wide variety of DOD organizations and agencies and several applied tasks have been carried out using it. The application of CAESAR II and progress in this task was reported in the last semi-annual report (June 1996).

Task 3 started in 1995. This task is focused on applying CAESAR II to the Adaptive Architectures for Command and Control (A2C2) program. The first effort in this task was reported separately in a progress report issued in July 1996. A summary description of the work documented in that report is contained in Section 4.2.

Task 4 addresses the graphical representation of C2 decision making using influence diagrams and the supporting Bayesian networks. The results of the first stage of this effort are presented in Section 4.3. Task 7, to be carried out in the Spring of 1997, addresses the implementation of these results as a module in CAESAR II.

Task 5 constituted the major theoretical effort during this reporting period. Its focus is the development of models and algorithms for distributed coordination in adaptive command and control teams. The results from this research effort are described in Section 4.1. The completed Ph.D. thesis of Didier M. Perdu will appear as a separate technical report in the spring of 1997.

Task 6 addresses the question of Measures - Measures of Performance and Measures of Effectiveness and their application to adaptive architectures for command and control. Earlier work has been reviewed and updated and a new module, using commercial software (MATLAB™) is being implemented in CAESAR II for use with the analysis and evaluation of the A2C2 experiments. Progress in this task is reported in section 4.4.

4.1 DISTRIBUTED PROCESS COORDINATION IN ADAPTIVE COMMAND AND CONTROL TEAM

The previous progress report has described the methodology for designing and evaluating adaptive command and control teams. The research effort during this period has focused on the evaluation part of the methodology. The complete methodology for adaptive team design is summarized in section 4.1.1 and is applied to an example in section 4.1.2.

4.1.1 Summary of the Methodology For Adaptive Team Design

The methodology for Adaptive Team design is summarized in Figure 4.1. It consists of the following steps:

Step 1. Construct the Process Model. This is done in the following substeps:

Step 1.1. Define the set of missions to be carried out by the team.

Step 1.2. For each mission defined in step 1.1, perform a functional decomposition up to the point where single functions can be performed by single team members in their intirety.

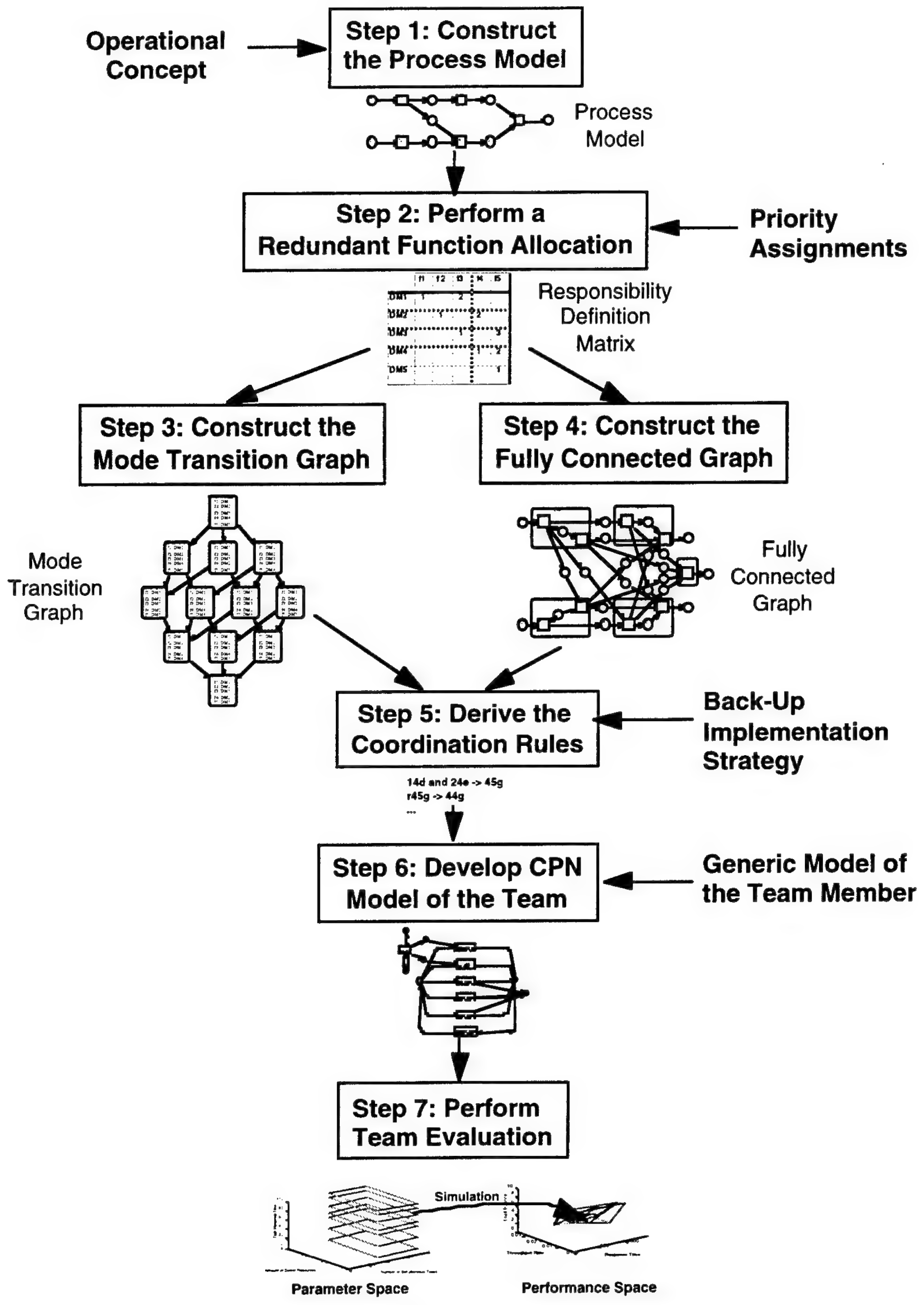

Operational Concept
Step 1: Construct the Process Model
Process Model
Step 2: Perform a Redundant Function Allocation
Priority Assignments
Responsibility Definition Matrix
Step 3: Construct the Mode Transition Graph
Step 4: Construct the Fully Connected Graph
Mode Transition Graph
Fully Connected Graph
Step 5: Derive the Coordination Rules
Back-Up Implementation Strategy
14d and 24e -> 45g
r45g -> 44g
...
Step 6: Develop CPN Model of the Team
Generic Model of the Team Member
Step 7: Perform Team Evaluation
Simulation
Parameter Space
Performance Space

Figure 4.1 Adaptive Team Design Methodology

Step 1.3. For each mission, construct the Process Model, which defines the data exchanged between the functions at the lowest level of the functional decomposition. This process model is represented by an Ordinary Petri Net where transitions represent the functions and the places represent the item of information exchanged between functions

Step 1.4. Fold together the different Process Models into the Functional Architecture which is represented by a Colored Petri Net. Check Lu's (1992) coordination constraints and modify the process models to correct identified problems.

Step 2. Perform a Redundant Function Allocation. For each mission and its Process Model, assign redundant responsibilities to team members: construct the Responsibility Definition Matrix which specifies the order in which the back-ups of each function by different team members will take place.

Step 3. Construct the Mode Transition Graph. For each mission, derive from the Responsibility Definition matrix the Mode Transition Graph, which defines the different Modes of Operation of the team activities and the transitions from one mode to another resulting from the back-up of a single function.

Step 4. Construct the Fully Connected Graph. For each mission, derive from the Responsibility Definition Matrix and the Process Model, the Fully Connected Graph which is a Petri Net Representation of all defined responsibility assignment and the required data exchanges between team members. Check that the Fully Connected Graph reduces to all the different modes of operation defined by the Mode Transition Graph.

Step 5. Derive the Coordination Rules. This is done in the following substeps:

Step 5.1. Determine the candidate back-up implementation strategies to be used by the team to transfer responsibilities.

Step 5.2. Derive the coordination rules for the Sender-Initated back-up implemetation strategy. This derivation is done in two substeps:

Step 5.2.1 Derive the execution rules. The Process Model determines the basic execution rules for each function. The Fully Connected Graph allows to make the correspondence between the left-hand side of the basic execution rules and the messages received by the team members. This correspondence allows to specify the different enablement conditions of the transitions of the Fully Connected Graph. The default output messages for each allocated function are derived from the active links of the Fully Connected Graph in the mode of operations with the lowest degree in which this allocated function is exercised

Step 5.2.2 Derive the transition rules. Construct the vector representations of the different modes of operations, using the label of the places of the Fully Connected Graph and the modes of the Mode Transition Graph to determine which places are active in a given mode of operations. For each team member, consider the restriction of these vectors to the places connected to the team member. For each arc in the Mode Transition Graph which connects mode A to Mode B, substract the vector representation of mode A from the vector representation of mode B. By making the distinction between places that are inputs to the team member, internals or outputs, derive the transition rules.

Step 5.3 Derive the coordination rules for the back-up implementation strategies defined in Step 5.1 by reallocating and modifying adequately the coordination rules derived in step 5.2.

Step 6. Develop the CPN Model of the Team. This is done in two substeps:

Step 6.1. Construct the upper level page of the model by drawing substitution transitions referring to the model of the team member for each team member. Connect these transitions to the a substitution transition referring to the model of the communication network. Specify appropriately which team members receive inputs from the environment and which ones produce the team output. Implement an input generator which provides appropriate inputs to the team.

Step 6.2. On each instance page of the team member model, specify the data specific to the corresponding team member by entering as initial markings of the places of color set *DMID* and *Rule* the dentification number of the team member and the translated rules.

Step 7. Perform Team Evaluation. This is done in the following substeps:

Step 7.1. Conduct Logical Evaluation. Use the validation and verification approach of Zaidi (1994) to identify logical problems in the rule base that can impede the coordination process.

Step 7.2. Conduct Behavioral Evaluation. Simulate the CPN of the adaptive team derived in Step 6 to make sure that the team behaves properly, that is, that the team output is produced and transfers of responsibility occur.

Step 7.3. Conduct Performance Evaluation. Identify the parameters and Measures of Performance of interest, instrument the CPN model of the adaptive team to collect data to evaluate MOPs. Construct the performance locus by simulating the model for selected points of the parameter locus. Compute the Measures of Effectiveness for a set of requirements. The value of the MOE for each candidate design is the basis for the selection of the final candidate design.

4.1.2 Application of the methodology to an example

To illustrate the methodology being described in the previous section, let us consider a fictitious example of an Air interdiction mission planning system. The Air Force generates every 24 hours the Air Tasking Order (ATO) which defines for every aircraft the mission it has to carry out two days later. The day in-between involves mission preparation. These missions can be either tactical missions, that is, aiming at one or several targets, or support missions, that is, to support the aircraft accomplishing tactical missions: refueling, air defense, ... The generation of the ATO is a complex process because it requires to coordinate in space and time the actions of resources of different types that are geographically distributed on different air bases. Generally, a subset of the resources is left at the disposal of the ground forces to support their actions. The missions of these aircraft kept in reserve can be of two types: Close Air Support and Air Interdiction. Close Air Support involves the destruction of enemy forces on the front line. These missions require special care to avoid fratricide and therefore can not be planned in the context of the Air Tasking Order. Air Interdiction aims at targets behind the front line to reduce as much as possible the support of the enemy forces: logistics support, reserves, ... Here again, these missions can not be performed in the context of the Air Tasking Order, because fast planning and action is required to be effective. The planning of these missions is performed by an Army Command and Control Team, which receives (1) real time information about enemy positions and friendly positions and (2) requests for air support. The Army C2 team has to do the detailed planning of the missions to satisfy these requests.

Step 1: Construct the Process Model

Five functions listed in Table 4.1 have been identified. The data being exchanged between functions are listed in Table 4.2. The first function, denoted by f1, is called "Analyze Request." It has for input a request denoted by the label "a" coming from units on the battlefield, or an intelligence report about some enemy movement behind the enemy lines. This function evaluates the threat of the enemy forces in the battlefield area designated by the request or the report and produces a Threat Report (denoted by "d"). Function f1 takes 50s to be executed. Function f2, "Get Enemy Data" assesses and evaluates the enemy position in the battlefield, especially in the area designated by the request or the intelligence report and generates an enemy posture report (denoted by "e"). 40 seconds are needed to perform this function. Function f3 performs the target development and prioritization, constructs the aimpoint and defines the best weapon to destroy the target. Taking the threat characteristics as input (coded by "c"), it generates the target data (denoted by "f") after a delay of 70 seconds. The fourth function denoted by f4 is called "Perform Penetration/Attrition Analysis." It forecasts the degree of redundancy necessary for the objective given the enemy posture, its air defense capability in the area of interest and the characteristics of the threat. It generates a Penetration/Attrition Analysis Report (coded by "g"). Function f4 is performed in 80 seconds. The last function, "Plan Mission" is denoted by f5 and delivers the final output of the team, "h". It combines the target data and the information contained in the penetration/analysis report to define completely the mission. The defined mission is denoted by "h". It takes 90 seconds to complete this mission planning.

Table 4.1 Functions of the Air Interdiction Planning System

Code	Description	Delay
f1	Analyze Request	50 s
f2	Get Enemy Data	40 s
f3	Perform Target Development/Prioritization and Aimpoint Construction & Weaponeering	70 s
f4	Perform Penetration/Attrition Analysis	80 s
f5	Plan Mission	90 s

Table 4.2 Data Exchanged

Code	Description
a	Request for CAS or Intelligence Report
b	Enemy Position
c	Threat Characteristics
d	Threat Report
e	Enemy Posture
f	Target Data
g	Penetration/Attrition Analysis
h	Planned Mission

The process model derived from this description of the functions is shown on Figure 4.2. Inputs from the environment, a and b, are processed respectively by functions f1 and f2. The output of f1, c, is processed by function f3, while f4 needs d, the other output of f1, and e, the output of f2, to be performed. Function f5 needs the results f and g of functions f3 and f4 to produce the team output, h.

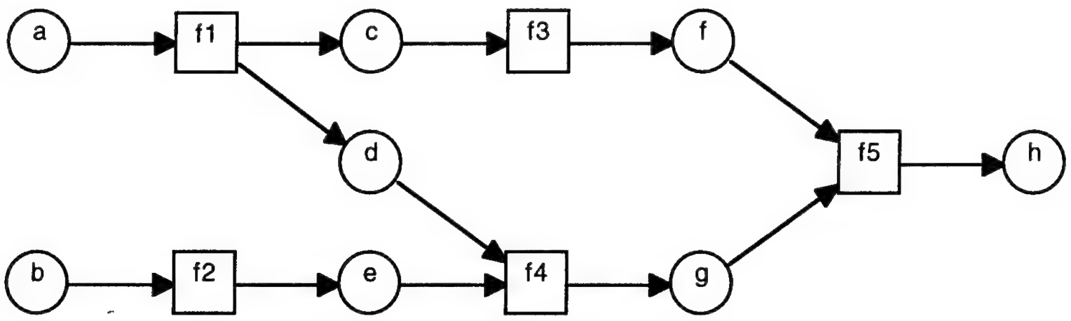


Figure 4.2 Petri Net Representation of a Process Model

Step 2: Perform a Redundant function Allocation

Let us consider that the mission has to be carried out by a team of 5 members: DM1, DM2, DM3, DM4 and DM5. The responsibility definition matrix of Table 4.3 shows the order in which the back-up of each function is implemented. The rows of the matrix represent the different team members, while the columns represent the various functions. The matrix elements represent the priority of assignments with 1 denoting the baseline allocation, 2 the primary back-up, etc. There is no back-up envisioned for f1 and f2, which are respectively performed by DM1 and DM2. The reason is that other team members would need to get the same inputs from the environment to perform this back-up. DM3, DM4 and DM5 are responsible, respectively, for functions f3, f4 and f5 in the normal mode of operation. In a back-up mode, DM1 can perform f3 and DM2 can perform f4. Finally, the back-up of f5 is performed first by DM4 and then by DM3.

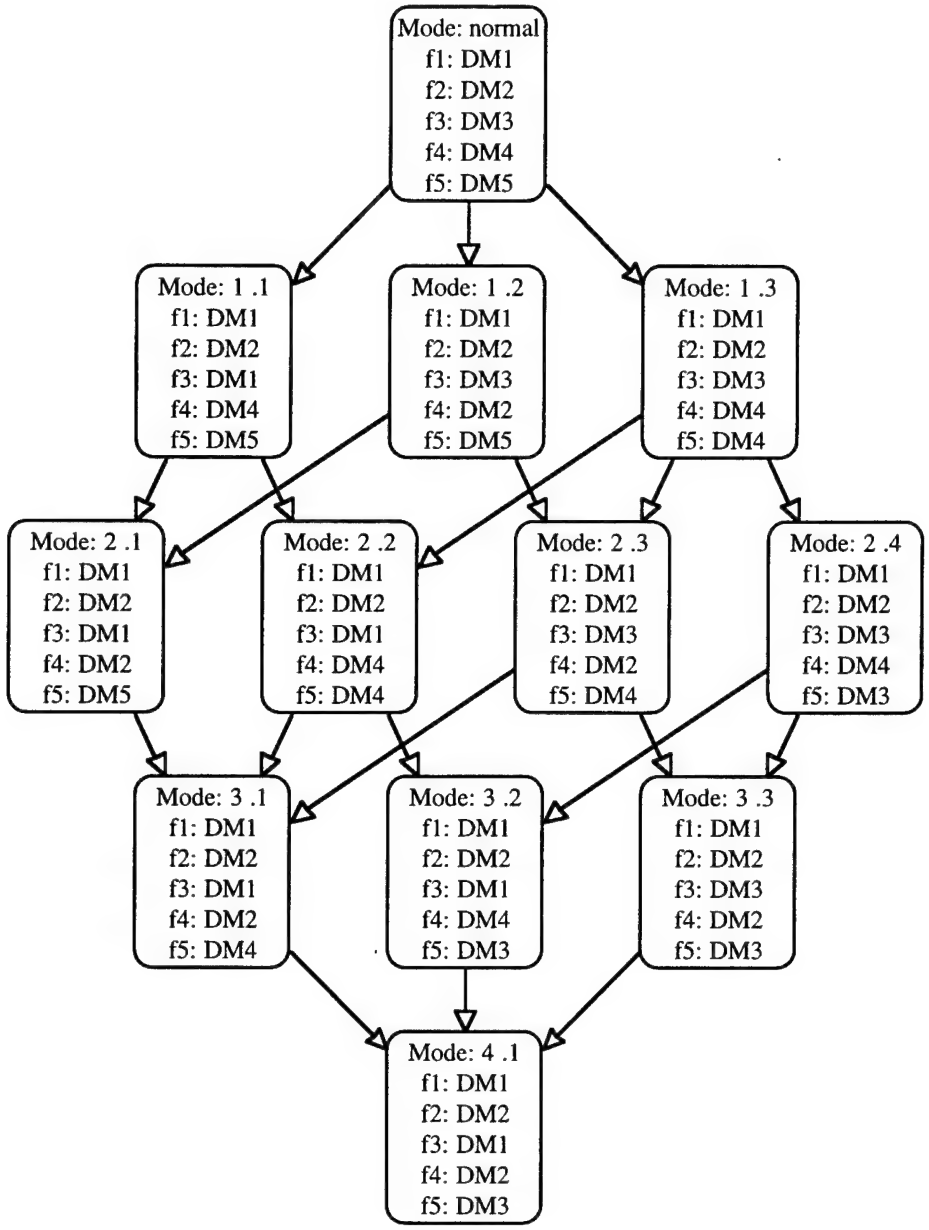
Table 4.3 Responsibility Definition Matrix

	f1	f2	f3	f4	f5
DM1	1		2		
DM2		1		2	
DM3			1		3
DM4				1	2
DM5					1

Step 3: Construct the Mode Transition Graph

A mode of operation is defined as the allocation of each function to each team member. The order of back-up implementation for each function introduces a partial ordering in these modes of operation. It can be shown that the set of the modes of operations is a lattice. The Hasse diagram of this lattice is called the Mode Transition Graph.

Figure 4.3 shows the mode transition graph for the example. Each mode has been assigned a label X.Y where X indicates the number of back-up operations that have occurred in the mode of operations and Y enumerates the modes having the same number of back-ups.



Mode: normal f1: DM1 f2: DM2 f3: DM3 f4: DM4 f5: DM5
Mode: 1 .1 f1: DM1 f2: DM2 f3: DM1 f4: DM4 f5: DM5
Mode: 1 .2 f1: DM1 f2: DM2 f3: DM3 f4: DM2 f5: DM5
Mode: 1 .3 f1: DM1 f2: DM2 f3: DM3 f4: DM4 f5: DM4
Mode: 2 .1 f1: DM1 f2: DM2 f3: DM1 f4: DM2 f5: DM5
Mode: 2 .2 f1: DM1 f2: DM2 f3: DM1 f4: DM4 f5: DM4
Mode: 2 .3 f1: DM1 f2: DM2 f3: DM3 f4: DM2 f5: DM4
Mode: 2 .4 f1: DM1 f2: DM2 f3: DM3 f4: DM4 f5: DM3
Mode: 3 .1 f1: DM1 f2: DM2 f3: DM1 f4: DM2 f5: DM4
Mode: 3 .2 f1: DM1 f2: DM2 f3: DM1 f4: DM4 f5: DM3
Mode: 3 .3 f1: DM1 f2: DM2 f3: DM3 f4: DM2 f5: DM3
Mode: 4 .1 f1: DM1 f2: DM2 f3: DM1 f4: DM2 f5: DM3

Figure 4.3 Mode Transition Graph for the Example

Step 4: Construct the Fully Connected Graph

The Fully Connected Graph represents the complete allocation of functions to team members and the exchange of data between them for the execution of these functions. It is constructed algorithmically from the Responsibility Definition Matrix and the Process Model. Figure 4.4 shows the Fully Connected Graph for the example.

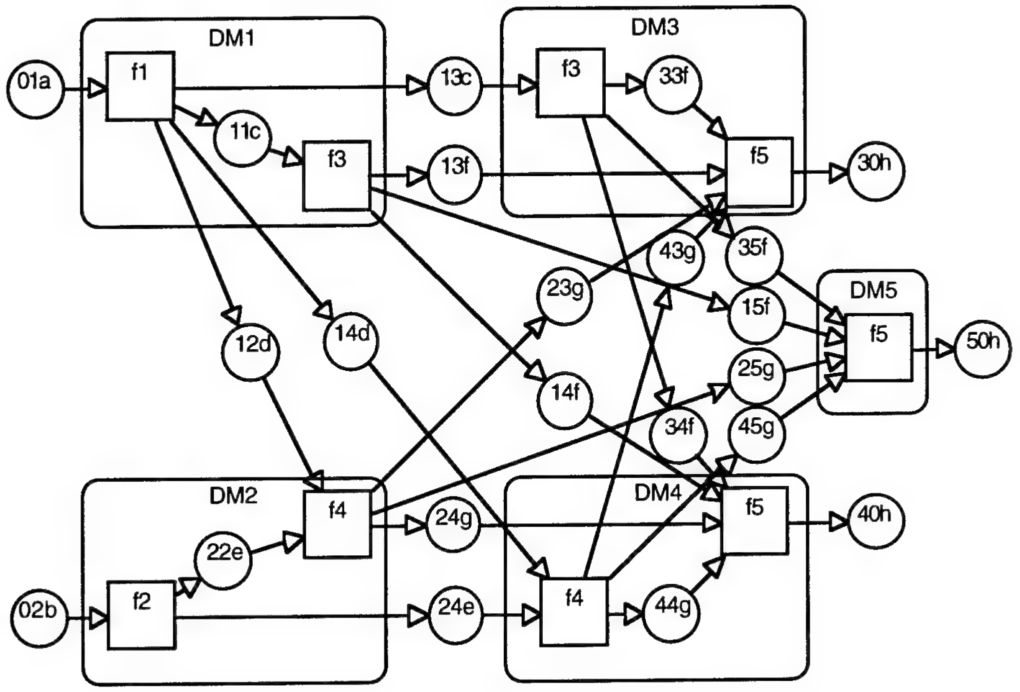


Figure 4.4 Fully Connected Graph for the Example

Step 5: Derive the Coordination Rules

The coordination rules are derived from the Mode Transition Graph and the Fully Connected Graph. The coordination rules depend on the back-up implementation strategy used by the team to transfer responsibility. Two different back-up implementation strategies are considered for the example: the Sender-Initiated (SI) strategy and the Receiver-Initiated (RI) strategy. In the SI strategy, the delivery of the item of information to a team member able to use the piece of information to perform a function is the responsibility of the team member who generated this piece of information. Team members receiving a message containing the piece of information may refuse it, if they are overloaded. In the RI strategy, the responsibility for implementing the back-up of a function belongs to the team member receiving a message and unable to perform the related function. When this occurs, the team member has to forward the corresponding message to the team member able to perform the back-up as specified in the responsibility definition matrix. All subsequent messages related to the refused tasks has to be forwarded to this team member.

The previous report described in details how to generate the coordination rules for the sender-initated strategy. Table 4.4 shows the rules for the SI strategy for the example. The rules for other back-up implementation strategies can be derived from the rules for the SI strategy by appropriately transferring rules from one team member to another and changing the labels of the messages. The reason is that the rules for the SI strategy identify the different anomalies that can occur. The rules for the RI strategy are derived from the SI strategy rules as follows.

First, the SI strategy rules that initiate the back-up of a function by the emitter of the message remain unchanged. This is to avoid the useless transmission of information back and forth between two team members. It is assumed that the team members keep in memory the information they send to other team members in case they have to perform the back-up of the corresponding function. For the other rules, to assign the responsibility of initiating the back-up to the recipient of the messages, the rules of the type rX $\rightarrow$ Y needs to be transferred to the team member who generates the message rX. The rules are then simplified. The next step is to change in the first field of the messages in all the rules to reflect which team member is sending the corresponding message. After this substitution process, duplicates of the same rule need to be eliminated.

In the example, in the SI strategy, the initiation of the back-up of f5 from DM5 to DM4 is done through the following rules:

DM2: r25g $\rightarrow$ 24g

DM3: r35f $\rightarrow$ 34f

DM4: 44g & 34f $\rightarrow$ 40h

24g & 34f $\rightarrow$ 40h

DM5: 35f & overl $\rightarrow$ r35f & nof5

35f & nof5 $\rightarrow$ r35f & nof5

25g & overl $\rightarrow$ r25g & nof5

25g & nof5 $\rightarrow$ r25g & nof5

The rules of DM2 and DM3 are transferred to DM5 and, after simplification of the rules, DM5 has the following rule:

35f & overl $\rightarrow$ 34f & nof5

35f & nof5 $\rightarrow$ 34f & nof5

25g & overl $\rightarrow$ 24g & nof5

25g & nof5 $\rightarrow$ 24g & nof5

To reflect the fact that the item f is sent from DM5 to DM4, the predicate 34f in the rules of DM4 and DM5 needs to be changed into 54f. For the same reason, 24g has to be replaced by 54g in the different rules. As a result, the receiver-initiated strategy rules are:

DM2: -

DM3: -

DM4: 44g & 54f $\rightarrow$ 40h

54g & 54f $\rightarrow$ 40h

DM5: 35f & overl $\rightarrow$ 54f & nof5

35f & nof5 $\rightarrow$ 54f & nof5

25g & overl $\rightarrow$ 24g & nof5

25g & nof5 $\rightarrow$ 24g & nof5

The total set of rules for the RI strategy is displayed on Table 4.5.

Table 4.4 Rules for the Example when the Sender-Initiated Strategy is Used

DM1	01a → 13c & 14d	Execution of f1. c is sent to DM3, d is sent to DM4
	r13c → 11c	Item c refused by DM3 is kept in memory
	11c → 15f	Execution of f3. Output f is sent to DM5
	r15f → 14f	Item f refused by DM5 is sent to DM4
	r14f → 13f	Item f refused by DM4 is sent to DM3
	r14d → 12d	Item d refused by DM4 is sent to DM2
DM2	02b → 24e	Execution of f2. Output e is sent to DM4
	r24e → 22e	Item e refused by DM4 is kept in memory
	22e & 12d → 25g	Execution of f4. Output g is sent to DM5
	r25g → 24g	Item g refused by DM5 is sent to DM4
	r24g → 23g	Item g refused by DM4 is sent to DM3
DM3	13c → 35f	Execution of f3. Output f is sent to DM5
	r35f → 34f	Item f refused by DM5 is sent to DM4
	r34f → 33f	Item f refused by DM4 is kept in memory
	33f & 43g → 30h	Execution of f5
	33f & 23g →30h	
	13f & 43g → 30h	
	13f & 23g →30h	
	13c & overl → r13c	Item c from DM1 is refused if overloaded
DM4	14d & 24e → 45g	Execution of f4. Output g is sent to DM5
	r45g → 44g	Item g refused by DM5 is kept in memory
	44g & 34f → 40h	Execution of f5
	44g & 14f → 40h	
	24g & 34f → 40h	
	24g & 14f → 40h	
	14d & overl → r14d	Item d from DM1 is refused if overloaded
	24e & overl → r24e	Item e from DM2 is refused if overloaded
	14d & nof4 → r14d & nof4	Item d from DM1 is refused if e was refused
	24e & nof4 → r24e & nof4	Item e from DM2 is refused if d was refused
	34f & overl → r34f & nof5	Item f from DM3 is refused if overloaded
	14f & overl → r14f & nof5	Item f from DM1 is refused if overloaded
	24g & overl → r24g & nof5	Item g from DM2 is refused if overloaded
	44g & nof5 → 43g & nof5	Item g is sent to DM3 item f was refused
	34f & nof5 → r34f & nof5	Item f from DM3 is refused if item g was refused
	14f & nof5 → r14f & nof5	Item f from DM1 is refused if item g was refused
	24g & nof5 → r24g & nof5	Item g from DM2 is refused if item f was refused
DM5	35f & 45g → 50h	Execution of f5
	15f & 45g → 50h	
	35f & 25g → 50h	
	15f & 25g → 50h	
	35f & overl → r35f & nof5	Item f from DM3 is refused if overloaded
	15f & overl → r15f & nof5	Item f from DM1 is refused if overloaded
	25g & overl → r25g & nof5	Item g from DM2 is refused if overloaded
	45g & overl → r45g & nof5	Item g from DM4 is refused if overloaded
	35f & nof5 → r35f & nof5	Item f from DM3 is refused if item g was refused
	15f & nof5 → r15f & nof5	Item f from DM1 is refused if item g was refused
	25g & nof5 → r25g & nof5	Item g from DM2 is refused if item f was refused
	45g & nof5 → r45g & nof5	Item g from DM4 is refused if item f was refused

Table 4.5 Rules for the Example when the Receiver-Initiated Strategy is Used

DM1	01a → 13c & 14d	Execution of f1. c is sent to DM3, d is sent to DM4
	r13c → 11c	Item c refused by DM3 is kept in memory
	11c → 15f	Execution of f3. Output f is sent to DM5
DM2	02b → 24e	Execution of f2. Output e is sent to DM4
	r24e → 22e	Item e refused by DM4 is kept in memory
	22e & 42d → 25g	Execution of f4. Output g is sent to DM5
DM3	13c → 35f	Execution of f3. Output f is sent to DM5
	43f & 43g → 30h	Execution of f5
	13c & overl → r13c	Item c from DM1 is refused if overloaded
DM4	14d & 24e → 45g	Execution of f4. Output g is sent to DM5
	54g & 54f → 40h	Execution of f5
	44g & 54f → 40h	
	14d & overl → 42d	Item d from DM1 is sent to DM2 if overloaded
	24e & overl → r24e	Item e from DM2 is refused if overloaded
	14d & nof4 → 42d & nof4	Item d from DM1 is sent to DM2 if e was refused
	24e & nof4 → r24e & nof4	Item e from DM2 is refused if d was refused
	54f & overl → 43f & nof5	Item f from DM5 is sent to DM3 if overloaded
	54g & overl → 43g & nof5	Item g from DM5 is sent to DM3 if overloaded
	r45g → 44g	Item g refused by DM5 is kept in memory
	44g & nof5 → 43g & nof5	Item g is sent to DM3 if item f was refused
	54f & nof5 → 43f & nof5	Item f from DM5 is sent to DM3 if g was refused
	54g & nof5 → 43g & nof5	Item g from DM5 is sent to DM3 if f was refused
DM5	35f & 45g → 50h	Execution of f5
	15f & 45g → 50h	
	35f & 25g → 50h	
	15f & 25g → 50h	
	35f & overl → 54f & nof5	Item f from DM3 is sent to DM4 if overloaded
	15f & overl → 54f & nof5	Item f from DM1 is sent to DM4 if overloaded
	25g & overl → 54g & nof5	Item g from DM2 is sent to DM4 if overloaded
	45g & overl → r45g & nof5	Item g from DM4 is refused if overloaded
	35f & nof5 → 54f & nof5	Item f from DM3 is sent to DM4 if g was refused
	15f & nof5 → 54f & nof5	Item f from DM1 is sent to DM4 if g was refused
	25g & nof5 → 54g & nof5	Item g from DM2 is sent to DM4 if f was refused
	45g & nof5 → r45g & nof5	Item g from DM4 is refused if f was refused

Step 6: Develop the CPN Model of the Adaptive Team

The previous report described a generic model of the team member. This model is used to construct the model of adaptive team using Hierarchical Colored Petri nets. In Hierarchical Colored Petri Nets, substitution transitions can replace subnets that are drawn on other subpages and different substitution transitions can then refer to the same subpage, leading to the creation of different instances of the same subpage. Each instance behaves independently of the others. The model of the adaptive team, shown on Figure 4.5, is then constructed by drawing five substitution transitions corresponding to the five team members. They all refer to the same page "DM#3" that contains the model of the team member described in the previous report. The five team members are connected by a communication network as represented by the substitution transition "Comms," through two places of color set *Msg*. The subtitution transition *Comms* refers to the model page "Comms#4", described in the previous report. DM3, DM4 and DM5 can produce the team output and they are connected to the sink place of color set *Data* on the right hand side of the model. DM1 and DM2 can receive inputs from the environment and their substitution transitions have one input socket place of color set *Data*. The transition *scenario* generates tokens in these places that represent data from the environment. The place *TaskID* defines the ID of the tasks. The place *Delay* defines the interarrival time of the tasks. The guard function of the transition *scenario* indicates the total number of tasks to be processed.

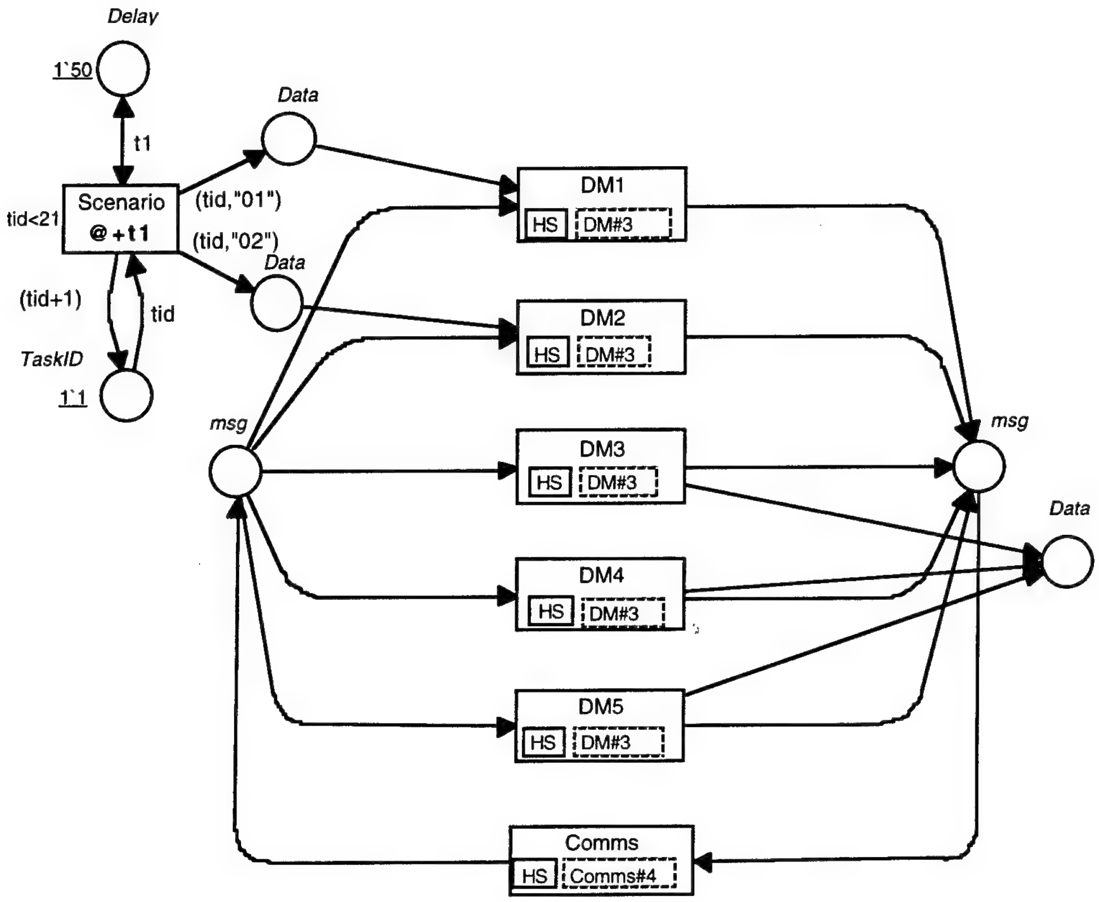


Figure 4.5 Model of the Team for the Example

The specificities of each instances of the team member model, that is their ID numbers and the rules allocated to each of them are specified inside each instance as initial markings of the places of color set *DMID* and *Rule*. To accomplish that, the rules derived in Step 5 needs to be translated into tokens of the color set "rule". A rule of the type:

$p_1 \;\&\; p_2 \;\&\; \ldots \;\&\; p_n \rightarrow q_1 \;\&\; q_2 \;\&\; \ldots \;\&\; q_m$

which takes a delay t to be executed is represented by the token of color set "Rule":

1\`(["$p_1$", "$p_2$", .... "$p_n$"], \`(["q_1", "q_2", ... "q_m"], t)

A certain syntax for these tokens is required for the model to behave correctly. Conclusions in the right hand side of the rules that refer to messages sent to other team members needs to be modified as follows. A prefix is added to specify what is the type of the message:

- "s" for an information message,
- "n" for a refusal message,
- "f" for a forced information message that can not be refused by the addressee, and
- "a" for an acknowledgment message.

The ID number of the team member to which the message is addressed is appended as a suffix. For instance, in the example, the fact r13c generated by DM3 to indicate that he refuses message 13c is translated into "nr13c1".

For the modeling of a team member being overloaded, the left hand sides of the rules that are of the form X & overl are translated into "refX". "refX" is generated when the transition "Receive msg" fires and when the number of tasks currently being processed is larger than the maximal number. In the example, the statement 13c & overl indicating that DM3 is overloaded when receiving the item 13c is represented by the string "ref13c".

To trigger the updating of the list of the tasks being processed, the fact "done" to indicate that the processing of a task is complete for the time being needs to be added in the right hand side of the rules which result in a message being sent or a team output being produced.

Facts on the right hand of the rules referring to team output produced by the team member are modified by adding the prefix "o" to the string predicate. For instance, in the example, 30h, indicating that the output of the team h is produced by DM3, is translated into the string "o30h".

Additional facts can be added to keep track of what and when certain rules have fired. for example, the fact that DM3 performs f5 in back-up can be recorded by adding the string "3didf5" in the second list of the token rule: the rule 33f & 43g -> 30h, which takes 90 seconds to be executed, is translated into the token"

```
        1`(["33f", "43g"], ["o30h", "3didf5","done"], 90)
```

Finally, to keep track of the tasks being currently processed or waiting to be processed, execution rules need to split into two subrules. The first subrule models the fact that all the data necessary for the execution of a function are present in the working memory of the team member and does not take any delay. The execution of the first subrule results in the list of current tasks being or to be processed by the team member modeled by the list token in the place of color set "TaskList" being updated. The second subrule is the execution of the function and takes the delay associated with the function to execute. In the example, the rule 33f & 43g -> 30h that models function f5 performed by DM3 is translated into two tokens of the color set "Rule":

```
        1`(["33f", "43g"], ["3dof5"],0)+1`(["3dof5"], ["o30h", "3didf5","done"], 90)
```

Table 4.6 shows the translation of the rules used in the example for the SI strategy. The right column corresponds to the initial marking of the places of color set "Rule" of each instance of the model corresponding to each team member DM1 to DM5.

Table 4.6 Rules Translation for the SI Rules

DM1	01a → 13c & 14d	1`(["01a"],["1dof1"],0)+
		1`(["1dof1"],["s13c3", "s14d4", "done"], 50)+
	r13c → 11c	1`(["r13c"], ["11c"], 0)+
	11c → 15f	1`(["11c"], ["1dof3"],0)+
	r15f → 14f	1`(["1dof3"],["s15f5","1didf3", "done"], 70)+
		1`(["r15f"], ["s14f4", "done"], 0)+
	r14f → 13f	1`(["r14f"], ["f13f3", "done"], 0)+
	r14d → 12d	1`(["r14d"], ["f12d2", "done"], 0)
DM2	02b → 24e	1`(["02b"], ["2dof2"],0)+
		1`(["2dof2"],["s24e4", "done"], 40)+
	r24e → 22e	1`(["r24e"], ["22e"], 0)+
	22e & 12d → 25g	1`(["22e","12d"], ["2dof4"],0)+
		1`(["2dof4"],["s25g5","2didf4","done"], 80)+
	r25g → 24g	1`(["r25g"], ["s24g4", "done"], 0)+
	r24g → 23g	1`(["r24g"], ["f23g3", "done"], 0)
DM3	13c → 35f	1`(["13c"], ["3dof3"],0)+
		1`(["3dof3"],["s35f5", "done"], 70)+
	r35f → 34f	1`(["r35f"], ["s34f4", "done"], 0)+
	r34f → 33f	1`(["r34f"], ["33f"], 0)+
	33f & 43g → 30h	1`(["33f", "43g"], ["3dof5"],0)+
	33f & 23g → 30h	1`(["33f", "23g"], ["3dof5"],0)+
	13f & 43g → 30h	1`(["13f", "43g"], ["3dof5"],0)+
	13f & 23g → 30h	1`(["13f", "23g"], ["3dof5"],0)+
		1`(["3dof5"],["o30h", "3didf5"","done], 90)+
	13c & overl → r13c	1`(["ref13c"], ["nr13c1", "done"], 0)
DM4	14d & 24e → 45g	1`(["14d", "24e"], ["4dof4"],0)+
		1`(["4dof4"],["s45g5", "done"], 80)+
	r45g → 44g	1`(["r45g"], ["44g"], 0)+
	44g & 34f → 40h	1`(["44g", "34f"], ["4dof5"],0)+
	44g & 14f → 40h	1`(["44g", "14f"], ["4dof5"],0)+
	24g & 34f → 40h	1`(["24g", "34f"], ["4dof5"],0)+
	24g & 14f → 40h	1`(["24g", "14f"], ["4dof5"],0)+
		1`(["4dof5"],["o40h", "4didf5","done"], 90)+
	14d & overl → r14d	1`(["ref14d", "overl"], ["nr14d1", "done"], 0)+
	24e & overl → r24e	1`(["ref24e", "overl"], ["nr24e2", "done"], 0)+
	14d & nof4 → r14d & nof4	1`(["14d", "nof4"], ["nr14d1","nof4","done"], 0)+
	24e & nof4 → r24e & nof4	1`(["24e", "nof4"], ["nr24e2","nof4","done"], 0)+
	34f & overl → r34f & nof5	1`(["ref34f"], ["nr34f3", "nof5", "done"], 0)+
	14f & overl → r14f & nof5	1`(["ref14f"], ["nr14f1", "nof5", "done"], 0)+
	24g & overl → r24g & nof5	1`(["ref24g"], ["nr24g2", "nof5", "done"], 0)+
	44g & nof5 → 43g & nof5	1`(["44g", "nof5"], ["f43g3", "nof5", "done"], 0)+
	34f & nof5 → r34f & nof5	1`(["34f", "nof5"], ["nr34f3","nof5","done"], 0)+
	14f & nof5 → r14f & nof5	1`(["14f", "nof5"], ["nr14f1","nof5","done"], 0)+
	24g & nof5 → r24g & nof5	1`(["24g", "nof5"], ["nr24g2","nof5","done"], 0)
DM5	35f & 45g → 50h	1`(["35f", "45g"], ["5dof5"],0)+
	15f & 45g → 50h	1`(["15f", "45g"], ["5dof5"],0)+
	35f & 25g → 50h	1`(["35f", "25g"], ["5dof5"],0)+
	15f & 25g → 50h	1`(["15f", "25g"], ["5dof5"],0)+
		1`(["5dof5"],["50h", "done"], 90)+
	35f & overl → r35f & nof5	1`(["ref35f"], ["nr35f3", "nof5", "done"], 0)+
	15f & overl → r15f & nof5	1`(["ref15f"], ["nr15f1", "nof5", "done"], 0)+
	25g & overl → r25g & nof5	1`(["ref25g"], ["nr25g2", "nof5", "done"], 0)+
	45g & overl → r45g & nof5	1`(["ref45g"], ["nr45g4", "nof5", "done"], 0)+
	35f & nof5 → r35f & nof5	1`(["35f", "nof5"], ["nr35f3","nof5","done"], 0)+
	15f & nof5 → r15f & nof5	1`(["15f", "nof5"], ["nr15f1","nof5","done"], 0)+
	25g & nof5 → r25g & nof5	1`(["25g", "nof5"], ["nr25g2","nof5","done"], 0)+
	45g & nof5 → r45g & nof5	1`(["45g", "nof5"], ["nr45g4","nof5","done"], 0)

Step 7: Perform Team Evaluation

Logical Evaluation was performed using RULER, the program developed by Zaidi (1994) to validate and verify decision rule base. The rules for each of the strategies were used as inputs to the program. The only problem identified by RULER was the presence of sources, different from the inputs, for the predicates "overl" that were making many rules incomplete. This problem is not a logical one because the value of the predicates "overl" (the team member is too busy or not) is determined by the current state of the system when these rules are triggered.

Behavior and Performance Evaluation were conducted simultaneously to assess how behavioral problems mentioned in the previous sections affect team performance. A scenario of 20 tasks was defined and used to simulate three candidate team designs: an adaptive team using the sender-initiated strategy, an adaptive team using the receiver-initiated strategy, and a team with fixed structure.

The team with a fixed structure used the same Colored Petri Net model as the adaptive teams. The rules allocated to the different team members are listed in Table 4.7. All the messages exchanged between team members are "forced" messages to prohibit the refusal of tasks by the team members.

Table 4.7 Rules for the Fixed Structure

DM1	01a → 13c & 14d	1`(["01a"],["1dof1"],0)+ 1`(["1dof1"],["f13c3", "f14d4", "done"], 50)
DM2	02b → 24e	1`(["02b"], ["2dof2"],0)+ 1`(["2dof2"],["f24e4", "done"], 40)
DM3	13c → 35f	1`(["13c"], ["3dof3"],0)+ 1`(["3dof3"],["f35f5", "done"], 70)
DM4	14d & 24e → 45g	1`(["14d", "24e"], ["4dof4"],0)+ 1`(["4dof4"],["f45g5", "done"], 80)
DM5	35f & 45g → 50h	1`(["35f", "45g"], ["5dof5"],0)+ 1`(["5dof5"],["50h", "done"], 90)

Three main parameters that have a large influence on team performance have been identified and varied: (1) the task interarrival time, (2) the amount of communication resources and, (3) the number of tasks each team member can process simultaneously. The rate at which tasks are submitted to the team is the main stressor to the team. The more frequent the inputs, the larger the number of tasks waiting to be processed and the more frequent transfers of responsibility have to take place. This parameter is expressed in seconds and was varied from 1 to 100 seconds. The second parameter is the amount of communication resources. Transfer of responsibilities requires the exchange of coordination messages between team members. The amount of communication resources available to support this information exchange will affect greatly the response time of the system. It corresponds to the number of channels available or amount of bandwidth. Two settings of communication resources were considered: 6 and 1. Six communication resources represent a nominal situation while 1 represents a case when communication resources are very scarce. The last parameter is the number of tasks that can be processed simultaneously by the team members A team member is able to process one or more tasks simultaneously. This number will have a large impact on team performance but also on the process of team adaptation. By decreasing this number, say from two to one, one can study the effect of fatigue of the team members on team performance. This parameter was set to 2 in the nominal case or 1 in the degraded case when the team members are tired. As a result, the parameter space is a cube in a three dimension space as shown on Figure 4.6

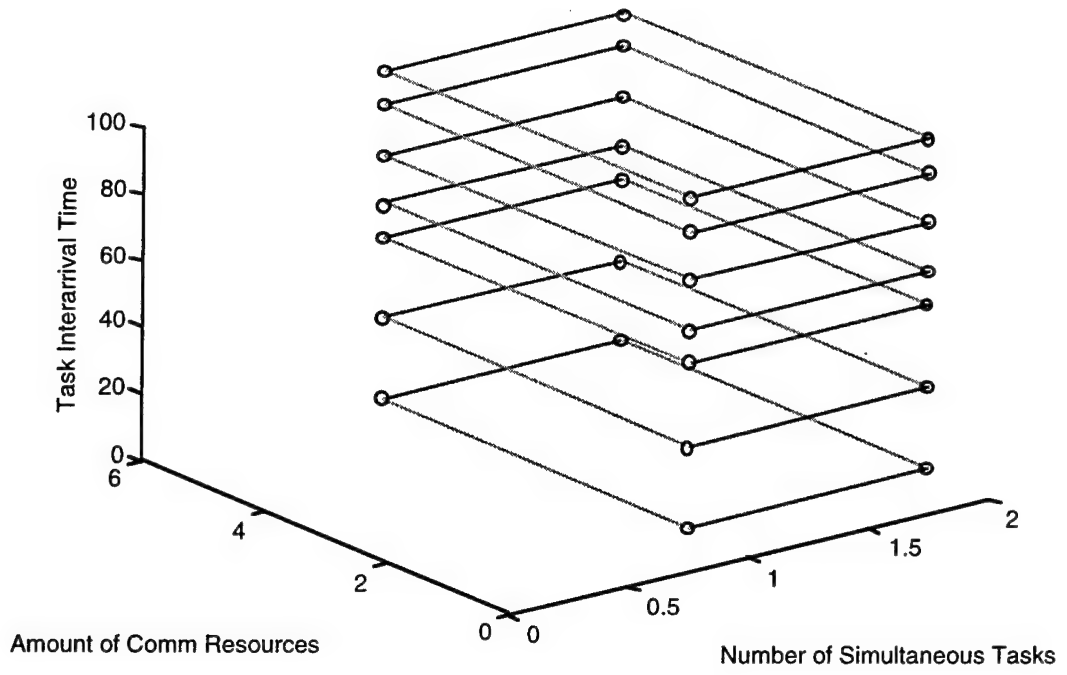


Figure 4.6 Parameter Space

Four Measures of Performance were evaluated: response time, throughput rate, average number of back-ups, and load balance.

Response Time

Response time is the average delay taken by the team to process an input. It is evaluated by recording for each input, its time of entry into the system, and the time when the related output is produced, subtracting the two times and averaging on the number of inputs.

Throughput Rate

Throughput rate is equal to the number of tasks processed per units of time. It is evaluated by recording the times when outputs are produced. It is important to note that since tasks can be performed out of sequence, the i-th output may not correspond to the i-th input. Let n be the number of inputs to be processed by the team and t_i be the time at which the i-th output is produced. The average interdeparture time D is equal to:

$$D = \frac{\sum_{i=1}^{n-1}(t_{i+1} - t_i)}{n-1} = \frac{t_n - t_1}{n-1}$$

The throughput rate R is then the inverse of D:

$$R = \frac{1}{D} = \frac{n-1}{t_n - t_1}$$

Average number of functions being backed-up

The average number of functions being backed-up is a measure of adaptation that characterizes the activity of an adaptive team. This measure is more a characterization of the behavior of the team than a Measure of Performance. In a first approach it is hard to assess whether more back-ups is better or worse than fewer back-ups. However, if the ordering of the back-up of functions done in the Responsibility Definition Matrix is defined according to the level of expertise that each team member has to accomplish each function, we can assume that backing-up of a function introduces a degradation in the quality of the execution of this function: the team member responsible for the back-up a function may perform worse than the team member initiating the back-up because he has less expertise to perform this function. In this case, the average number of functions being backed-up is related to the accuracy of the organization and the smaller the number of back-ups, the better the team performs. Accuracy is not considered here, because it depends extensively on the ability of the team to deal with the uncertain environment, which is an aspect of the team decision making process not addressed here.

Load Balance

Transfer of responsibilities is triggered when a team member is overloaded. A measure of load balance can be introduced to assess how well the total load is distributed among the team members. Let f_i (i = 1, ..., m) be the functions to be performed by the team for a task T_j and τ_i the delay it takes to perform f_i. We can relate the load to the total amount of time it takes to process one task, regardless or which team member performs which function. The load to process the single task T_j is therefore:

$$T_j = \sum_{i=1}^{m} \tau_i$$

If n is the number of tasks to be processed by the team, and if we assume that the delay of the functions do not vary with the tasks, the total load L is:

$$L = \sum_{j=1}^{n} T_j = n \sum_{i=1}^{m} \tau_i$$

let the team consist of ndm team members and let n_{ik} be the number of times the k-th team member has to perform function f_i for the n tasks to be processed by the team. If for each task, each function is performed exactly once, we have:

$$\sum_{k=1}^{ndm} n_{ik} = n$$

The load of the k-th team member, L_k, is equal to the amount of time it took to perform n_{ik} times each f_i:

$$L_k = \sum_{i=1}^{n} n_{ik} \tau_i$$

and we have:

$$\sum_{k=1}^{ndm} L_k = L$$

The measure of load balance, LB, is defined as the standard deviation of the ratio of L_k over L:

$$LB = \sigma(\frac{L_k}{L}) = \sqrt{E(\frac{L_k}{L} - E(\frac{L_k}{L}))^2}$$

The average of the load $E(L_k)$ is:

$$E(L_k) = \frac{L}{ndm} \text{ so that } E(\frac{L_k}{L}) = \frac{1}{ndm}$$

and corresponds to the case where the load L is equally distributed among the ndm team members. LB is therefore:

$$LB = \sqrt{E(\frac{L_k}{L} - \frac{1}{ndm})^2} = \sqrt{\frac{1}{ndm}(\sum_{k=1}^{ndm}(\frac{\sum_{i=1}^{m} n_{ik}t_i}{n\sum_{i=1}^{m} t_i} - \frac{1}{ndm})^2)}$$

LB measures how far the normalized load distribution is from a uniformaly distributed load $(\frac{1}{ndm})$. The lower the value of LB, the better the load is distributed among team members.

As an example, consider the extreme case of a team of ndm members where one team members, say DM1, performs all the functions for all the tasks. In that case, $L_1 = L$ and $L_k = 0$ for k = 2, 3, ..., ndm and LB is then expressed as:

$$LB = \sqrt{E(\frac{L_k}{L} - \frac{1}{ndm})^2} = \sqrt{\frac{1}{ndm}((1 - \frac{1}{ndm})^2 + (ndm - 1)(\frac{1}{ndm})^2)}$$

which reduces to:

$$LB = \sqrt{\frac{1}{ndm}\frac{(ndm - 1)^2 + ndm - 1}{ndm^2}} = \sqrt{\frac{ndm - 1}{ndm^2}}$$

For ndm = 5, the LB value attained for the extreme case where one team member does all the work is 40 %.

Let us now consider the case where all the functions have exactly the same delay τ and that each function is performed by a single team member for all the tasks, that is:

$ndm = m$ $\quad\quad$ $n_{ii} = n$ (i = 1, 2, ... , m) and $n_{ij} = 0$ if i j.

The total load L is then:

$$L = n \, . \, m \, . \, \tau$$

and the load of the k-th team member, L_k, is:

$$L_k = n\tau = \frac{L}{m} = \frac{L}{ndm}$$

The load is equally distributed among the team members and LB = 0.

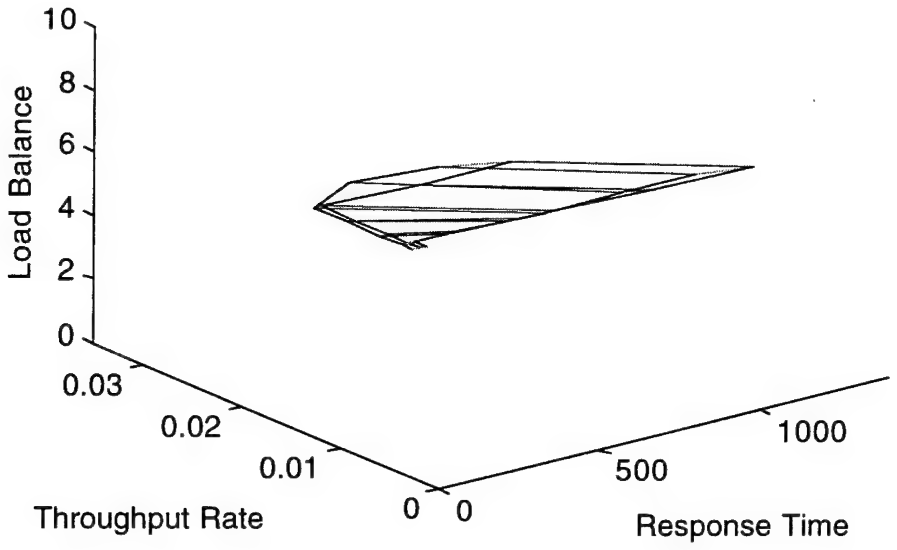
The model of the team was instrumented to collect the data allowing to compute the measures of performance: a sink place was added in the model page of the team member to record how many times each team member performed each function. The timed tokens contained in the output place of the team model at the end of a simulation specify when the output of each task was produced. A spreadsheet in Microsoft Excel was constructed to derive from these raw data the different Measures of Performance of interest attained for each simulation.

Table 4.8 shows the range of Measures of Performance attained by each candidate design. The projection of the Performance Locus on the space Response time x Throughput Rate x Load Balance and on the plane Response Time x Throughput Rates for each of the candidate designs is shown on Figures 4.7 to 4.9.

Table 4.8 Range of Measures of Performance attained by each Candidate Design

Candidate Design	Response Time (seconds)	Throughput Rate (Tasks / seconds)	Load Balance (%)	Average Number of Back-Ups
Fixed	226 - 1277	0.0097 - 0.0225	5.62 - 5.62	0 - 0
SI	226 - 1298	0.0092 - 0.0217	2.12 - 7.53	0 - 2.15
RI	226 - 1159	0.0089 - 0.0240	1.82 - 7.43	0 - 2.2

For the fixed structure, since there is no back-up taking place, there is no change in the Load Balance: the performance space is contained in an horizontal located at LB = 5.62, as shown on Figure 4.7. The projection of that plane on the parallel plane Response Time x Throughput Rate is also shown on Figure 4.7. The performance loci of the adaptive teams are volumes as shown in the top part of Figures 4.8 and 4.9. Adaptive teams have the same lowest response time as the fixed structure teams. For long input interarrival time, adaptive teams do not transfer responsibilities and behave like the fixed structure team leading to the same response time of 226 seconds attained for 6 communication resources and 2 simultaneous tasks per team member. When back-up is taking place, there is a change in Load Balance. In some cases, SI 0 performs worse than the fixed team as far as response time and throughput rates are concerned. RI 0 exhibits a large variation in response time and seems to achieve overall better performance than the fixed structure team. Both RI 0 and SI 0 reach load balance levels that may be worse than the fixed team. Finally, the receiver-initiated back-up implementation strategy seems to lead to better performance than the sender-initiated strategy. This is due to the fact that the receiver-initiated strategy requires fewer exchanges of coordination messages than the sender-initiated strategy.



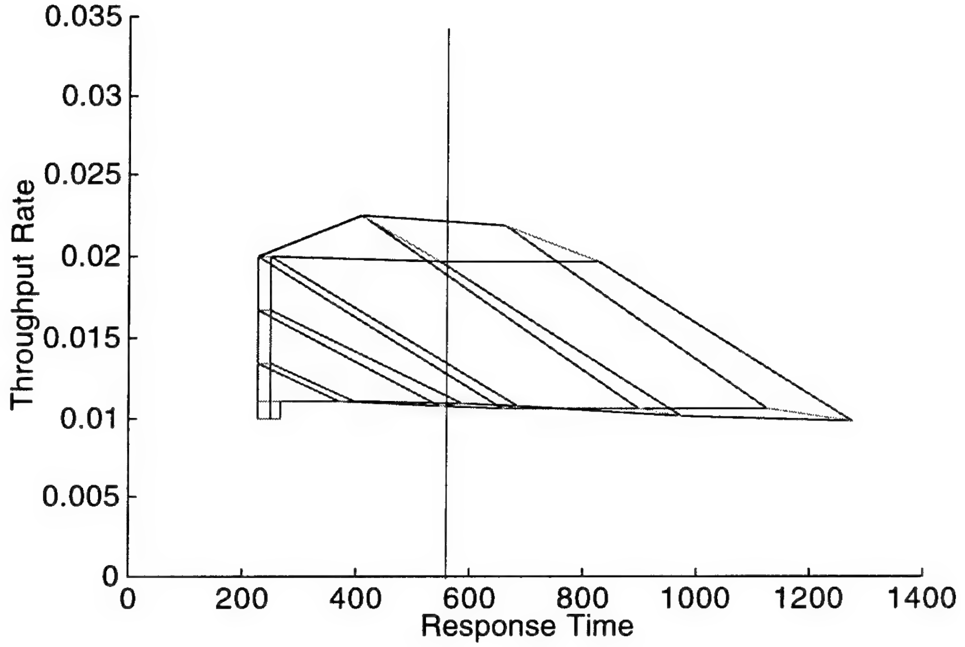
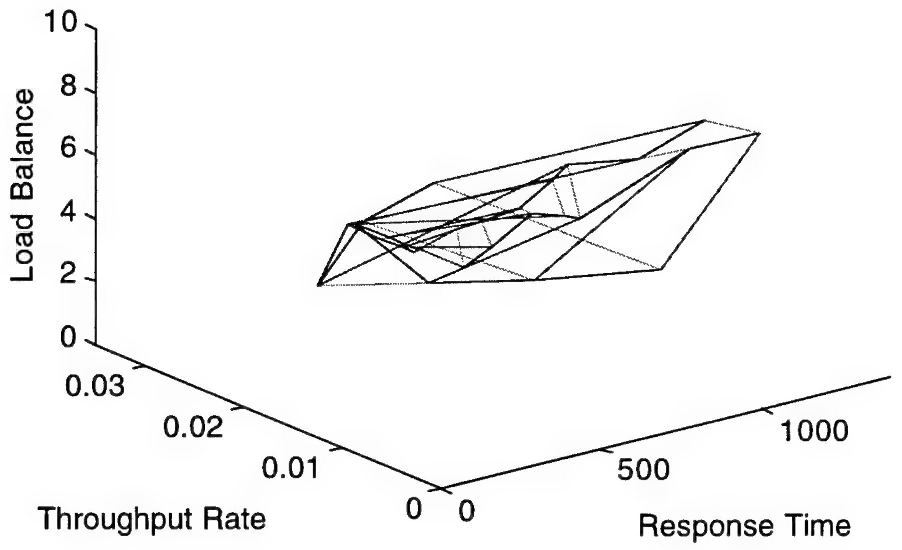


Figure 4.7 Performance Locus for Fixed



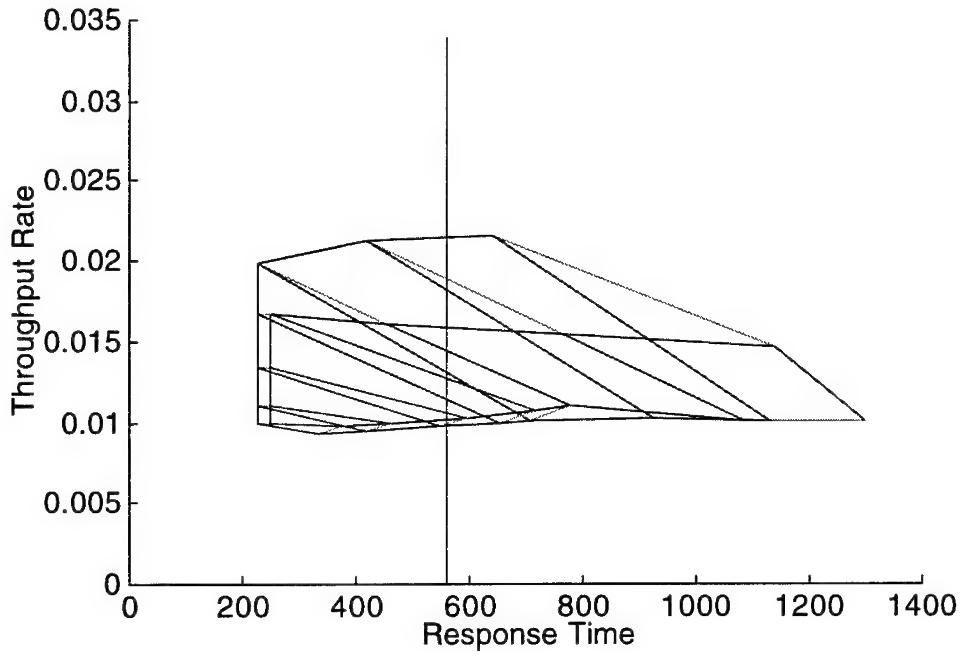
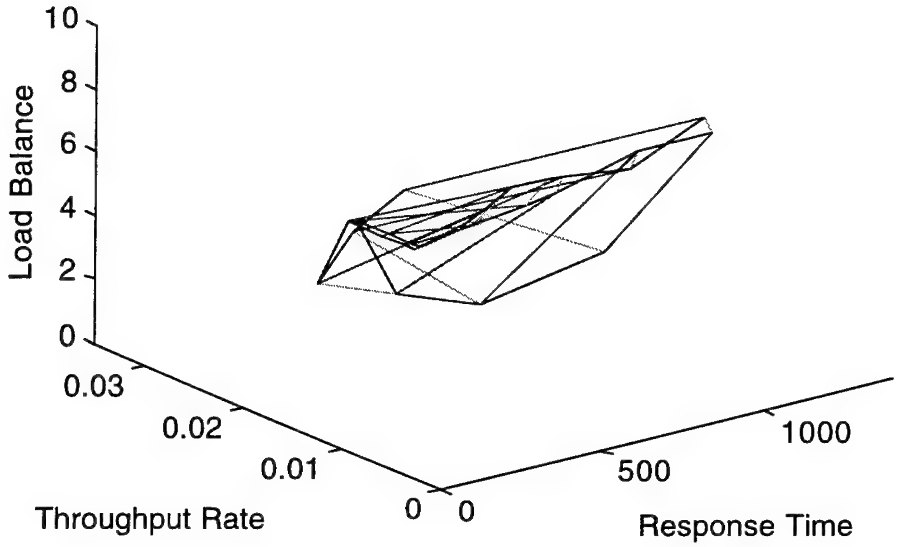


Figure 4.8 Performance Locus for SI 0



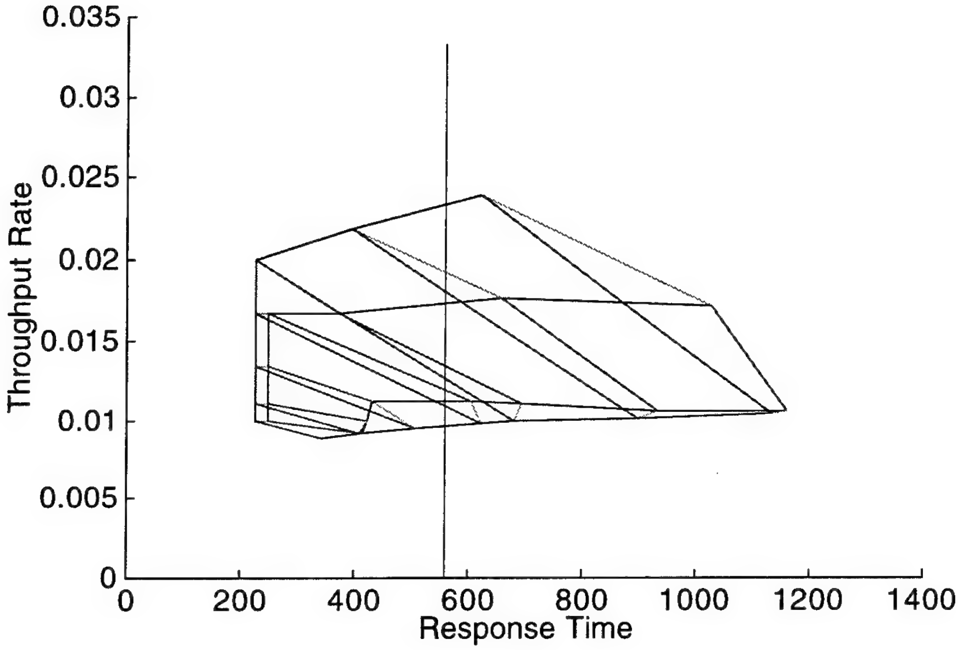


Figure 4.9 Performance Locus for RI 0

Let us assume, as performance requirements, that the mission has to be carried out in less than 10 minutes, and that no constraint is set for the other Measures of Performance. The effectiveness for each candidate team design can be measured by considering only the projection of the performance locus on the plane Response Time x Throughput Rate. The Measure of Effectiveness we have selected is equal to the ratio of the volume of the subset of the parameter locus that maps onto the part of the performance locus that resides inside the requirement locus over the total volume of the parameter locus. Let M denote the mapping from the parameter locus to the performance locus and M^{-1} the inverse mapping. Note that M^{-1} may not be a function if M is not a bijection, but can still be defined by looking at the results of the simulation. If L_a, L_p, L_r denote respectively the parameter, performance and requirements loci, we have:

$$\mathrm{MOE} = \frac{V(M^{-1}(L_p \cap L_r))}{V(M^{-1}(L_p))} = \frac{V(M^{-1}(L_p \cap L_r))}{V(L_a)}$$

Since L_a is rectangular parallelepiped in the parameter space, its volume, $V(L_a)$ can be easily computed. The next step is to identify $M^{-1}(L_p \cap L_r)$. Since only specific points of the parameter locus have been used to derive the performance locus, interpolation between some of these points is required to completely identify $M^{-1}(L_p \cap L_r)$. $M^{-1}(L_p \cap L_r)$ is then partitionned into parallelepipeds and tetrahedrons for which the volumes can be computed. Note that different partitions exist to cover $M^{-1}(L_p \cap L_r)$. The volume $V(M^{-1}(L_p \cap L_r))$ can then be approximated by the average of the sum of the volumes obtained for each partition. The Measures of Effectiveness obtained for the three candidate designs are listed in Table 4.9.

Table 4.9 MOE for the Five Candidate Designs

Candidate Design	Fixed	SI 0	RI 0
MOE	65 %	54 %	61 %

One can see from these results that the Fixed Structure Team is more effective than SI 0 and RI 0, SI 0 being much less effective than the other teams. These results show that the transfer of responsibilities between team members requires a communication network with enough resources to support the additional exchange of coordination messages required for the transfer to take place without degradation of performance.

A closer look at the behavior of the adaptive teams has led to the following observation. There seems to be an excessive number of back-ups that take place in cases when the interarrival time of the tasks is small. Transfer of responsibilities from one team member to another is triggered when a team member is busy when receiving the information items for a new task. This triggering process is instantaneous. Consider the case when a team member is almost done with a task when item of information related to a new task are received. The transfer of responsibility is then triggered for the function to be performed by another team member and items of information are forwarded to the other team member. When completing the older task the team member who initiated the responsibility transfer becomes idle, while the team member in charge of performing the back-up of the function for the new task has one more task to perform and thus most probably triggering additional transfer of responsibilities of his tasks to other team members. This "precipitation" of transfer of responsibility has thus a large influence on the response time of the team. A way to correct this undesirable behavior is to implement a buffer of tasks waiting to be executed for each team member. The transfer of responsibility is triggered when the buffer of waiting tasks is full. How the implementation of buffer increases the performance of an adaptive team is the interesting avenue of research which will be explored during the next period.

References

Lu Z. (1992) *Coordination in Distributed Intelligence Systems*. GMU/C3I-120-TH, M.S. Thesis, Center of Excellence in Command, Control, Communication, and Intelligence, George Mason University, Fairfax, VA.

Zaidi S. (1994). *Validation and Verification of Decision Making Rules.* Report GMU/C3I-155-TH. Ph. D. Dissertation, Center of Excellence in Command, Control, Communication, and Intelligence, George Mason University, Fairfax, VA.

4.2 ADAPTIVE ARCHITECTURES FOR COMMAND AND CONTROL (Levis and Perdu)

The A2C2 team developed a wargaming scenario as the basis of the first experiment conducted at the Naval Postgraduate School. The scenario was highly structured and was scripted in detail. At GMU, the scenario was analyzed and the embedded functions were identified. A structured analysis approach was used in the derivation of a discrete event model that realized the given scenario. First, the functions were organized in the form of a functional decomposition structure. This structure was used to develop an activity model in the form of IDEF0. Several iterations were necessary to resolve ambiguities in the scenario. From the IDEF0 model, an executable model was obtained in the form of a Colored Petri Net. The executable model was driven by the scenario variables and two key threads were identified. These key threads were traced in the IDEF0 diagram; a direct form of validating that the model represented indeed the scenario.

The whole process - the reverse engineering of the model from the scenario - highlighted the steps that need to be taken to define fully and unambiguously an architecture. In addition to the activity or process model, a data model needs to be defined that characterizes the information flows in the system; a rule model that contains the rules of the game - under what conditions are certain actions allowed; and a detailed data dictionary. Such a specification, when augmented with a set of state transition diagrams - would provide a rigorous basis for model driven experimentation.

The detailed description of this work - still exploratory in nature - has been documented in a separate report. It is continuing in preparation of analyzing Experiment 2 and it is driving the further development of CAESAR II.

4.3 MODELING ADAPTABILITY IN DYNAMIC DECISION NETWORKS (Buede and Wagenhals)

This section describes the continuation of Task 4: Graphical Representation of C2 Decision Making and Supporting Inference. In our previous effort, we used influence diagrams and Bayesian Networks to represent and examine the decision making process in hierarchical C2 organizations in which the decision making process is distributed and evolving over time. We illustrated the use of interconnected influence diagrams and Bayesian networks to create dynamic decision networks to represent the repetitive decision making tasks of formulating, instantiating, and adjusting plans over time as information becomes available about the status and effectiveness of friendly and enemy forces. Modeling the C2 decision making process at multiple levels of the organization allows the definition of what information needs to be communicated between levels and among peers at the same level. It also facilitates the definition and allocation of tasks to decision making entities within the overall hierarchical C2 organization. We also formulated a Petri Net model of the dynamic decision networks that addresses the key aspects of updating the tactical picture and making timely decisions (plan selection) on the basis of that tactical picture. The Petri Net is a discrete event model that captures the dynamic process of maintaining the tactical picture and the decision making processes that occur asynchronously.

One reason for creating a Petri Net model of C2 decision making is to analyze the logic, behavior, and performance of such systems. One of the most important concerns for the analysis of C2 systems is the ability of the system to adapt to changes in inputs, the environment, and the capabilities of the C2 system resources that are executing C2 tasks. In this context, a C2 system is a set of teams that carry out information processing and decision making in order to accomplish various missions. The representation of these processes as influence diagrams with embedded Bayesian networks can be used to define the pattern of interactions that must occur between team members to accomplish the distributed tasks and missions. Once these interactions are known, adaptation schemes can be developed and analyzed.

The influence diagram/Bayesian Network representation of the Dynamic Decision Network (DDN) captures one form of adaptation in that it reflects the C2 systems process for changing (adapting) plans or selecting alternative strategies or courses of action as the result of making inferences about the current and potential future states of the battlefield. This task adaptation is the desired output of any C2 system. C2 systems also need to be able to dynamically re-configure as changes occur either in the task workload on components of the C2 system or in the ability of the system components to accomplish in a timely manner the tasks that have been allocated to them. The primary way of providing for this adaptability is to add redundancy in the allocation of tasks to the physical components of the C2 system. This allows a component to take over the responsibility of other components that are unable to carry out their assigned tasks as required to maintain the performance of the C2 system. The DDN representation facilitates the logical definition of this redundancy.

In companion research to this effort (Task 5), Perdu and Levis have represented this redundancy with two basic types of back-up schemes; vertical and horizontal. Vertical back-up is defined as the transfer or devolution of responsibility for performing a task from one team member to another when the former is unable to perform any task for which he is responsible. This situation can occur if a C2 component is disabled, destroyed, or loses communications with the rest of the C2 system. Horizontal back-up is defined as the sharing of a functional responsibility between team members using a coordination scheme to adjust or balance the workload between the members to maintain overall performance. For any adaptive C2 system, these back-up schemes can be represented as a matrix that defines the primary and backup responsibilities of each team member or component of the C2 system. Each combination of responsibility that provides the correct functionality for the team represents a feasible mode of operation. The research also provides the procedure for developing an executable model of the C2 system that incorporates the user defined vertical and horizontal back-up schemes and the transition between them. Behavior and performance analysis can be accomplished with the executable model.

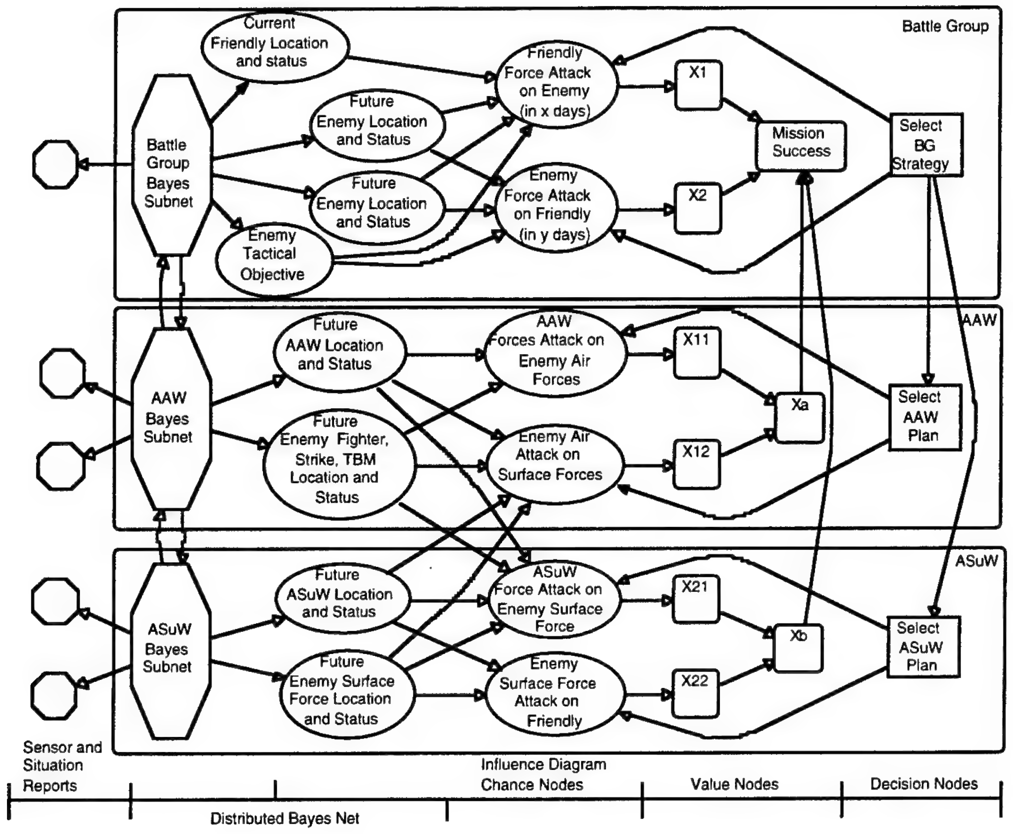

Battle Group
Current Friendly Location and status
Friendly Force Attack on Enemy (in x days)
X1
Future Enemy Location and Status
Mission Success
Select BG Strategy
Battle Group Bayes Subnet
Future Enemy Location and Status
Enemy Force Attack on Friendly (in y days)
X2
Enemy Tactical Objective
Future AAW Location and Status
AAW Forces Attack on Enemy Air Forces
X11
AAW
AAW Bayes Subnet
Xa
Select AAW Plan
Future Enemy Fighter, Strike, TBM Location and Status
Enemy Air Attack on Surface Forces
X12
ASuW
Future ASuW Location and Status
ASuW Force Attack on Enemy Surface Force
X21
ASuW Bayes Subnet
Xb
Select ASuW Plan
Future Enemy Surface Force Location and Status
Enemy Surface Force Attack on Friendly
X22
Sensor and Situation Reports
Influence Diagram Chance Nodes
Value Nodes
Decision Nodes
Distributed Bayes Net

Figure 4.10 Overall Influence Diagram with Imbedded Bayesian Net

The DDN representation described in this section can be used as a tool for developing the main inputs to the process of analyzing the design of adaptive teams. These inputs include the functional architecture and the responsibility matrix. We have chosen to represent the information processing and decision making of the C2 system as a DDN with two major components: a Bayesian net and an influence diagram. The Bayesian net "reads" inputs from sensors and situation reports from friendly forces as likelihoods and uses the net as an inference engine to calculate the posterior probabilities associated with current and future states of the world. Some of these probability distributions are used in the chance nodes of the influence diagram that represent each decision maker's decision problem of selecting the best strategy to pursue to achieve the mission goals at each level of the C2 system hierarchy.

Taken from our previous work, Figure 4.10 shows how the influence diagrams with embedded Bayesian nets could be allocated to physical nodes of the C2 system, represented as rounded boxes, and the interconnections between them. Figure 4.11 shows the allocation of the nodes of the Bayesian network to physical nodes of the C2 system. The arcs between the rounded boxes represent the interconnections needed. Figure 4.11 represents one mode of a possible set of modes of operation of an adaptable C2 System. It shows which physical C2 components or team members are responsible for updating specific nodes in the Bayesian net portion of the DDN.

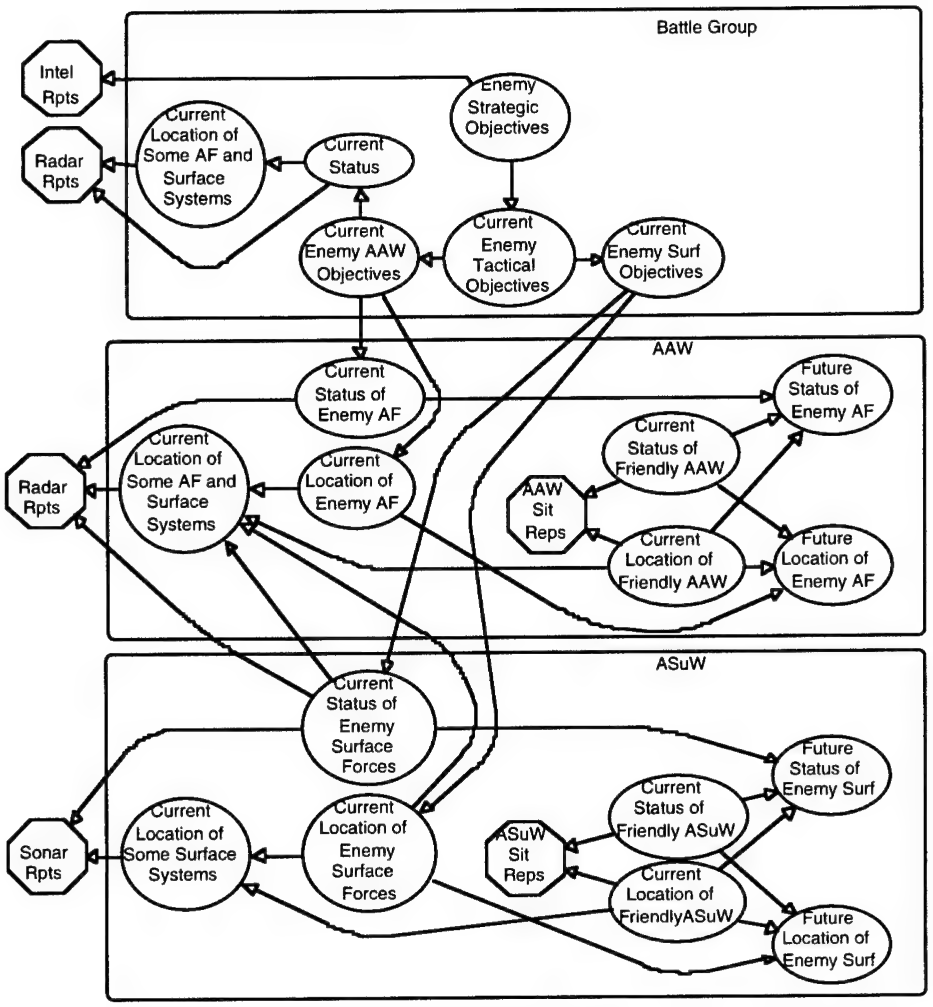

Battle Group
Intel Rpts
Radar Rpts
Current Location of Some AF and Surface Systems
Current Status
Enemy Strategic Objectives
Current Enemy AAW Objectives
Current Enemy Tactical Objectives
Current Enemy Surf Objectives
AAW
Current Status of Enemy AF
Future Status of Enemy AF
Current Location of Some AF and Surface Systems
Current Location of Enemy AF
Current Status of Friendly AAW
Radar Rpts
AAW Sit Reps
Current Location of Friendly AAW
Future Location of Enemy AF
ASuW
Current Status of Enemy Surface Forces
Future Status of Enemy Surf
Current Status of Friendly ASuW
Sonar Rpts
Current Location of Some Surface Systems
Current Location of Enemy Surface Forces
ASuW Sit Reps
Current Location of FriendlyASuW
Future Location of Enemy Surf

Figure 4.11 Allocation of Bayesian Network Nodes to Organizational Elements

Once the DDN has been constructed, a straight forward approach can be used to define the functions and their interconnections needed for updating the Bayesian net. The design of the adaptive C2 system can then be undertaken using the approach established by Perdu and Levis. We illustrate the procedure by first discussing the implementation of the algorithm that keeps the Bayesian net updated and then by showing how to use the DDN to develop the input to the team adaptability analysis technique of Perdu and Levis.

The Bayesian Net portion of the DDN acts as an inference engine by updating the probability distributions represented at each node of the network as new evidence in the form of sensor and intelligence reports arrives at the various nodes in the net. Each time a new observation arrives, a probability propagation algorithm is used to update the probabilities of the nodes based on that observation. To implement this inference engine, the algorithm is accomplished by a set of distributed activities (functions) that are interconnected by communications links in the some manner as the Bayesian net. Each activity uses an algorithm to create and send messages to its neighbors, receive messages from its neighbors, and update the probability distribution for which it is responsible. The most straight forward algorithm applies to Bayesian nets that are singly connected graphs. A graph is singly connected if there is no more than one chain between any two vertices. In such networks, the probability algorithm, attributed to Pearl, allows the network to be updated by having each activity compute λ values and messages to be sent from a child to its parent activities, π values and messages to be sent from a parent activity to each child, and computes the current value of the probability distribution of the node for which it is responsible.

A Petri Net representation of a portion of a singly connected Bayesian net and its corresponding interconnected functions that implement the propagation algorithm is shown in Figure 4.12 The figure shows two layers, the top layer contains the nodes of the Bayesian net with the arcs between the nodes shown with dotted lines to indicate they are not part of the Petri Net. The updating activity takes place over the Propagation Algorithm Layer implemented as a Petri Net. The dotted arcs and places at either end of the Petri Net are used to indicate that the net can be continued to accommodate additional Bayesian net nodes.

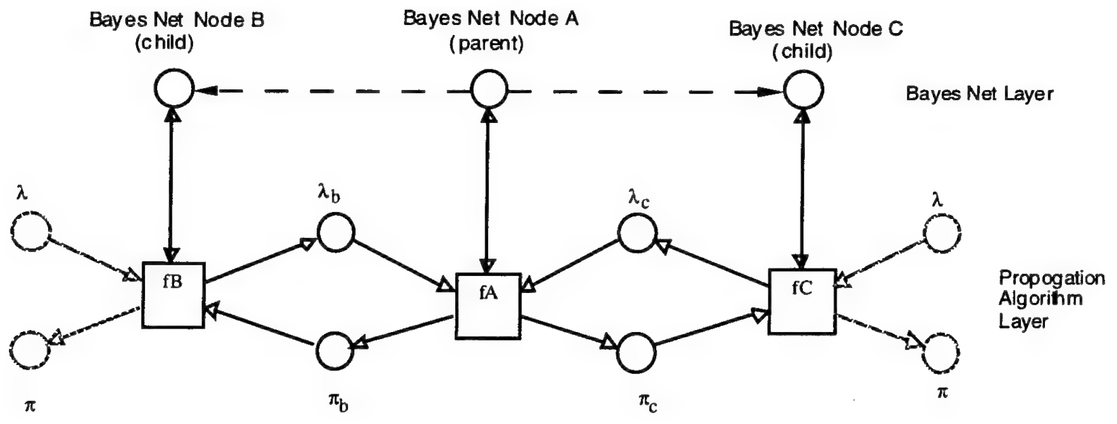


Figure 4.12 Petri Net representation of a singly connected Bayesian net

A portion of the Bayesian net of Figure 4.11 shown in Figure 4.13 will be used to illustrate how the DDN can be used to create an adaptable architecture. Note that the Bayesian net is singly connected. In the scheme for allocating nodes of Bayesian net to physical entities, certain nodes are duplicated between C2 physical nodes in the C2 system. One physical node would have the primary responsibility of updating the shared node and the other physical node would receive a copy of the value of the variable that node represents. For example, the Battle Group would have primary responsibility for Current Enemy Surface Objectives, and the ASuW would maintain a copy of that node.

The fact that certain decision making nodes maintain a copy of information updated by another node provides a logical method for defining backup redundancy for the C2 organization. For example, if the Battle Group of Figure 4.11 was unable to update the Current Enemy Surface Objectives node, a backup mode could be invoked with the ASuW updating that node. To implement this back-up mode, arrangements would have to be made for ASuW to have access to the Current Enemy Tactical Objectives node of the Battle Group during the period of back-up.

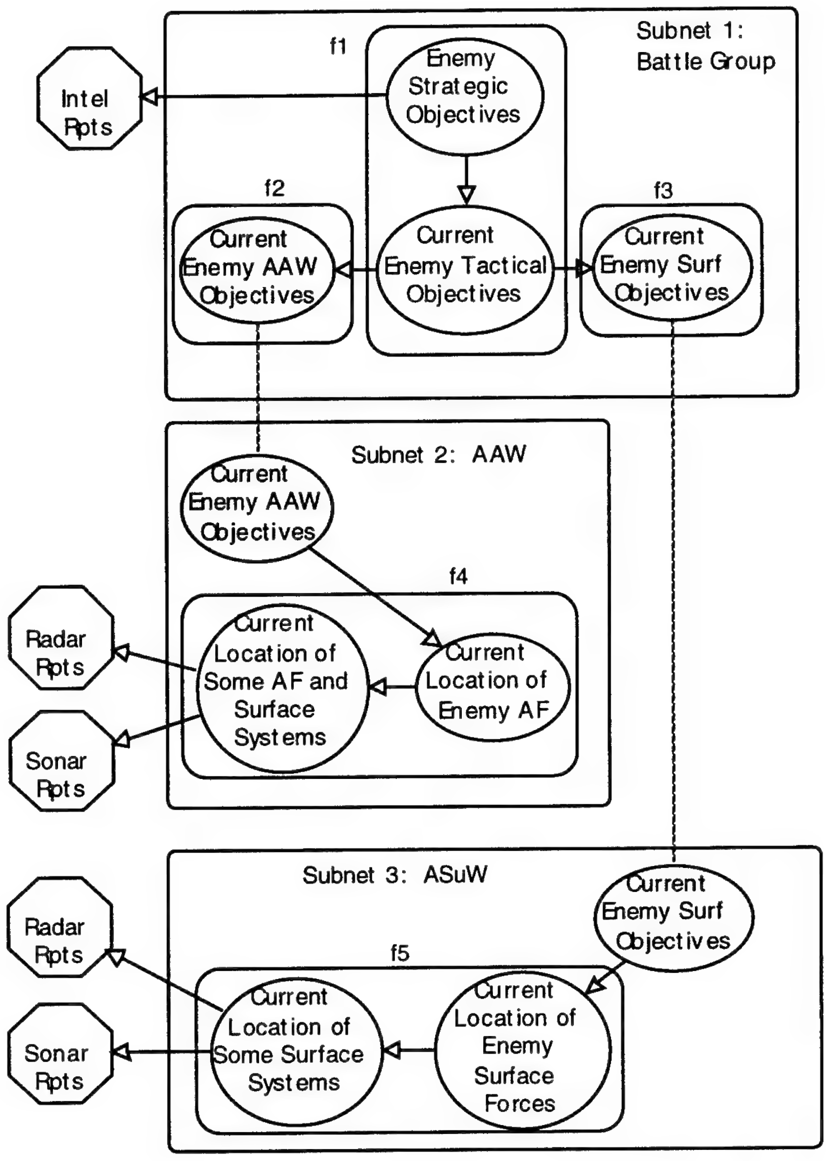

Subnet 1: Battle Group
f1
Intel Rpts
Enemy Strategic Objectives
f2
f3
Current Enemy AAW Objectives
Current Enemy Tactical Objectives
Current Enemy Surf Objectives
Subnet 2: AAW
Current Enemy AAW Objectives
f4
Radar Rpts
Sonar Rpts
Current Location of Some AF and Surface Systems
Current Location of Enemy AF
Subnet 3: ASuW
Current Enemy Surf Objectives
f5
Radar Rpts
Sonar Rpts
Current Location of Some Surface Systems
Current Location of Enemy Surface Forces

Figure 4.13 Subset of the Interconnected Bayesian Net

It is possible to develop a functional decomposition for the DDN by creating a correspondence between the nodes of the Bayesian net and the activities or function of updating those nodes. Functions are defined as an activity that updates nodes in the Bayesian net. The primary output of each function will be the updated value of the Bayesian net node for which it is responsible. These updating functions will be interconnected in the same topology as the Bayesian network and pass messages between each other according to the Bayesian net algorithm used. Five functions are indicated in Figure 4.13. Note that aggregation of functions can occur when two or more Bayes nodes are in a chain. Table 4.10 is a functional decomposition for the Bayesian net of Figure 4.13.

Table 4.10 Bayesian Net Updating Functions

Function	Bayesian Net Node(s) Update Activity
f1	Update Enemy Strategic Objectives and Enemy Tactical Objectives
f2	Update Current Enemy AAW Objectives
f3	Update Current Enemy Surface Objectives
f4	Update Current Location of Enemy Air Forces
f5	Update Current Location of Enemy Surface Forces

Using the functional decomposition and the inputs and outputs of each function as defined by the arcs of the DDN, a Petri Net representation of a Functional Architecture also can be derived as shown in Figure 4.14. For illustration purposes, the arcs of the Bayesian net are shown as dashed lines. They are not part of the Petri Net.

The next step is to allocate these functions to the physical C2 components in both primary and back-up schemes. Clearly, the original DDN indicates a primary scheme with the Battle Group responsible for f1, f2, and f3, AAW responsible for f4, and ASuW responsible for f5. The need to maintain duplicate copies of nodes suggests that it is logical for AAW to back-up f2 and ASuW to back-up f3 since they will already have a copy of the latest state of the node if a backup mode needs to be implemented. In addition, complete back-up for the Battle Group can be provided by assigning either AAW or ASuW back-up responsibility for f1. Then if the Battle Group becomes incapacitated or disconnected from AAW and ASuW, AAW will take over f1 and f2 and ASuW will take over f3. Additional back-up schemes could be considered such as having AAW can take over f5 from ASuW and ASuW take over f4 for AAW if required. For clarity, these are not considered in the illustration.

Once the primary and back-up function allocations have been defined using the DDN and the desired back-up concepts, the Responsibility Matrix can be easily derived. Table 4.11 shows the matrix for the simple example. Note that f8 and f9 are assumed to be assigned to the Radar and Sonar system, respectively and are not assigned to a decision making node.

Table 4.11 Responsibility Matrix

	Functions				
Decision Making Node	f1	f2	f3	f4	f5
Battle Group	1	1	1		
AAW	2	2		1	
ASuW			2		1

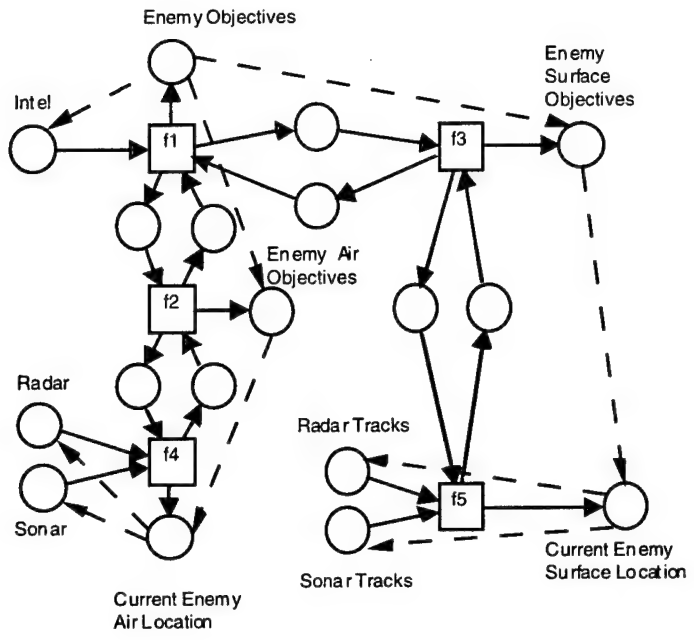


Figure 4.14 Functional Architecture of Bayesian Net

Given the Functional Architecture and the Responsibility Matrix it is possible to analytically derive the coordination scheme required to allow the organization to adapt to changing conditions using the techniques developed by Perdu and Levis. Indeed, for the above example, the algorithm developed using Design/CPN by Perdu has been used to create the Fully Connected Graph (Figure 4.15) and Mode Transition Graph (Figure 4.16) of the adaptive system. Details are contain in the Technical Report for the period 1 July 1995 to 30 June 1996.

In Figure 4.15, the primary mode of operation is shown by the bolded arcs. The back-up allocation of functions is shown with the lighter arcs and boxes. This scheme has 1 primary and 7 back-up modes as shown in the Mode Transition Graph. In the algorithm, rules are defined for the transition between various back-up modes. The various back-up modes are established by using the appropriate dotted arcs when the need for a back-up situation arises. Of course other back-up schemes could be defined by the user. The same techniques would be applied for each of them. Once this have been done, the behavior and performance analysis of the adaptive C2 system derived from the DDN representation can be accomplished using the techniques of Perdu and Levis.

Summary of Results:

We have shown how to use the Dynamic Distributed Network techniques of modeling the distributed decision making process of hierarchical C2 systems developed earlier to create a functional decomposition and functional architecture of a C2 system. The DDN can also be used to identify logical redundancies in the C2 system so that adaptation of the architecture can be examined. We illustrated the technique creating an input to the Design/CPN program of Perdu and Levis developed to analyze adaptability in C2 architectures.

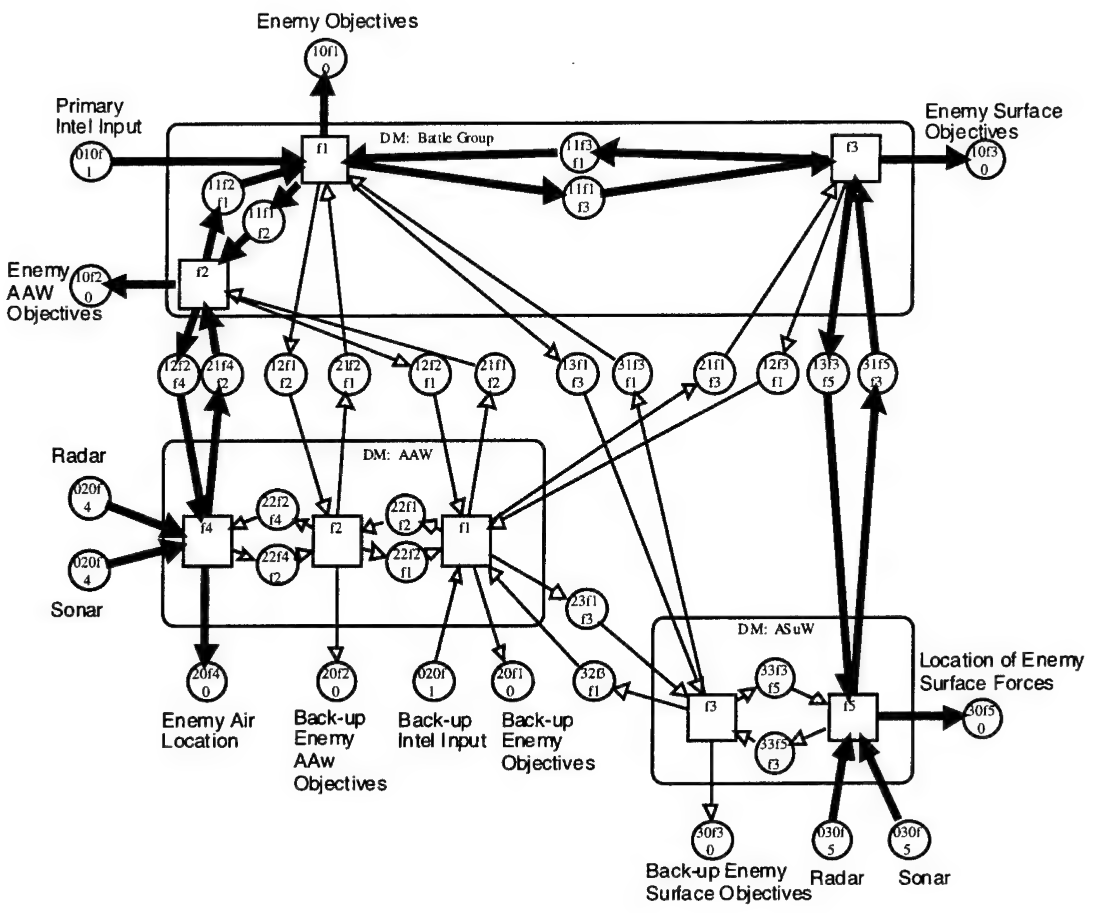


Figure 4.15 Fully Connected Graph of the Adaptive DDN

This work considered Bayesian nets that are singly connected. The example was for a singly connected tree, although the same technique will work for polytrees[1]. The techniques described in this work will have to be expanded in order to handle more complex Bayesian nets that include non-singly connected graphs. While the basic notion of message passing between functions still applies, non-singly connected networks require careful clustering of multiple nodes such as the creation of cliques of trees. Several techniques have been developed in which the goal is to find a representation that is a good approximation of the probability distribution, that is also computationally tractable. Once the tree of clusters has been selected, the techniques shown in this paper can be applied.

Reference

1988 Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inferences.* Morgan Kaufmann Publishers, 1988.

[1] A Polytree is a Directed Acyclic Graph that is singly connected (only one chain between any two nodes) where a node can have several parents. Results in tree-like structures with more than one root.

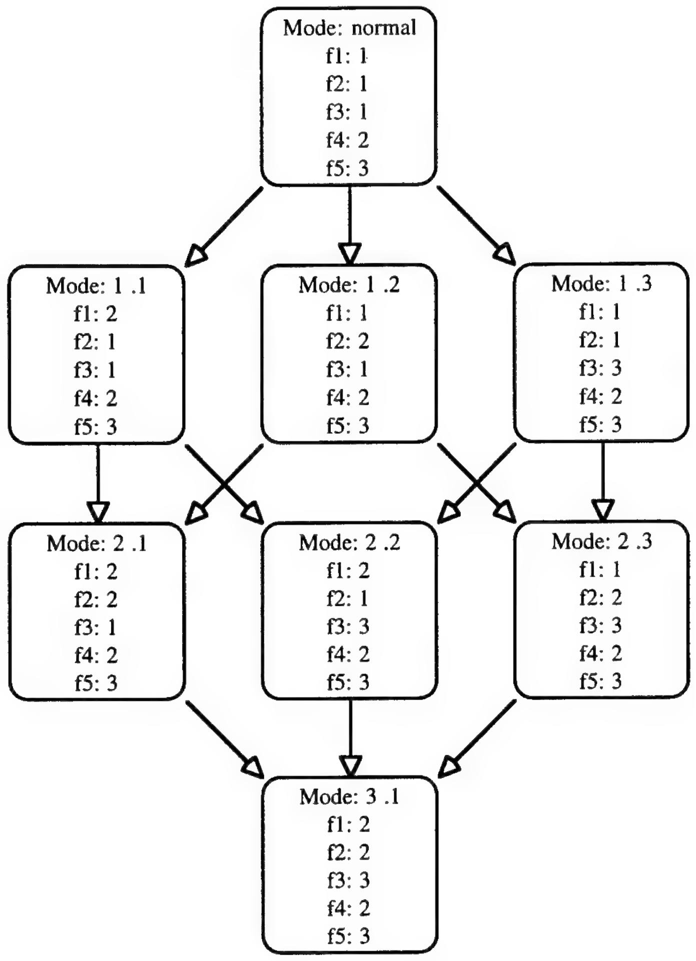


Figure 4.16 Mode Transition Graph Showing 8 Primary and Back-Up Modes

4.4 BEHAVIORAL AND PERFORMANCE EVALUATION OF ADAPTIVE ARCHITECTURES (Wagenhals, Perdu, Levis)

4.4.1 Introduction

This section describes the progress made on the design and development of the System Effectiveness Analysis Tool (SEAT) to support the evaluation of performance of adaptive architectures for decision making organizations. The tool is being developed as part of a suite of tools known as Computer Aided Evaluation of Systems ARchitectures (CAESAR II). SEAT is being designed so that the user does not have to be familiar with the software code to use the SEAT module. The module will have a user-friendly Graphical User Interface (GUI) so that given an input file of parameter and performance data, the module can be controlled to produce graphical plots. The plots that will be generated are Parameter Loci, Performance

Loci, Requirements Loci, and plots relating Measures of Effectiveness to changes in requirements.

The description is divided into four sections. Section 4.4.2 briefly describes CAESAR II and the performance evaluation concepts used in developing the evaluation module. Section 4.4.3 defines the requirements that have been established for SEAT. Section 4.4.4 details the design created to satisfy the requirements. Section 4.4.5 briefly describes the initial proof-of-concept testing that has been done with MATLAB® including the use of the GUI to generate output plots.

4.4.2. CAESAR II and Performance Evaluation

CAESAR II is a suite of Commercial Off-the-Shelf (COTS) software packages and specialized algorithms that constitute a toolbox that can be used for the analysis, design, and evaluation of architectures - architectures of decision making organizations as well as information architectures. It is based on a four stage framework: Requirements Analysis, System Analysis of static model views of the architecture, Synthesis of an executable model of the architecture, and Evaluation of the architecture. The SEAT software application will support the fourth stage of the CAESAR II framework, the purpose of which is to analyze and evaluate the behavior, performance, and effectiveness of the architectures. SEAT addresses the performance evaluation segment.

Four concepts are important to the performance evaluation process: Parameters, Measures of Performance (MOPs), Requirements, and Measures of Effectiveness (MOEs). A Parameter is a primitive concept that represents properties or characteristics of system entities. Parameters are sometimes called variables because they can assume a range of values. A value of a Parameter characterizes the behavior or structure of an entity. For example, processing time in seconds for a processor may can be a Parameter for that processor component of a system. In general, Parameters that are relevant to understanding behavior and performance of a system, can take on values over certain domains. The domain of a set of relevant Parameters can be characterized by a Parameter Locus in n dimension space where n is the number of Parameters. A point in the Parameter Space that is in the Parameter Locus represents an operating point of the system.

An MOP is a concept that quantifies attributes of system behavior. MOPs can be considered as functions of a set of parameters (e.g. MOP_i = f(parameter_1, parameter_2 . . . parameter_n). For example, the end-to-end time for a C3 system to process an input may be an attribute that quantifies one type of system behavior and thus is an MOP. It is calculated by subtracting the Parameter, time of arrival of an input (t_i), from the Parameter, time of the output (t_o). Thus MOP = (t_o - t_i). Its value is dependent upon the value taken by several Parameters that define the behavior of components of the C3 system. These are not necessarily the same Parameters as t_i and t_o. In general, a system can have many MOPs. Response time (or delay), throughput rate, accuracy, reliability, etc. are all MOP concepts used for C3 systems. As with the Parameter Locus, the set of relevant MOPs for a system can be characterized as a MOP or "Performance" Locus in m dimension space where m is the number of MOPs. Because the value taken by each MOP is dependent on the value of the set of Parameters relevant to that MOP, there is a mapping between the Parameter Locus in the Parameter Space and the MOP Locus in the Performance Space. The locus in the performance space characterizes the range of possible MOP values of the system over all the possible operating points defined in the Parameter Locus.

An MOE is a concept that quantifies how well a system performs its function. It implies the existence of a standard. These standards are determined from the Requirements of the system.

A Requirement is an attribute that expresses an aspect of the behavior or performance of a system that is significant in achieving a mission objective. Thus each Requirement can be specified by a variable whose value quantifies the value of the Requirement. Each Requirement is based on a formula or algorithm that uses Parameters for quantifying the requirement. Note that the Parameters used in the formula for the Requirements are not necessarily the same Parameters used in determining the value of the corresponding MOPs. An MOE is a formula or algorithm that is a function of the variables for measuring the performance (MOP) and the variables for measuring the Requirement variables (e.g. MOEj = f(MOP1, MOP2, . . . MOPm, Requirement_1, Requirement_2 . . . Requirement_m). Note that the MOP and the Requirement variables must be commensurate, that is they must have the same variable definitions and units. In general, a set of Requirements can be characterized as a locus in an m dimension Requirements Space where m is the number of Requirements variables. Since the MOP and Requirements variables must be commensurate, the Requirement Locus can be located in the same m dimension space as the MOP Locus.

As an example of the quantification of an MOE, consider two extremes. If the MOP Locus is totally within the Requirements Locus, then all system behavior meets the Requirements, and the system is 100% effective. On the other hand, if the MOP Locus is totally outside the Requirements Locus, then the system will never meet any of the Requirements under any operating condition, and the system effectiveness is zero. In general, systems fall in between these extremes, with only a portion of the MOP Locus falling inside the Requirements Locus. Two choices exist when this occurs. The first is to try to change the design of the architecture to cause the MOP Locus to fall within the Requirements Locus. The other choice is to review and possibly change the Requirements. Performing a MOP sensitivity analysis can aid in this evaluation. To perform this analysis, the value of the MOE, as a function of changes in the Requirement values, is computed.

An example of an MOE is the percentage of the MOP Locus that falls within the Requirement Locus,

$$\mathrm{MOE} = \frac{S(Lp \cap Lr)}{S(Lp)} \qquad (1)$$

where S(Lp $\cap$ Lr) is a measure (area, volume, etc.) of the intersection of Lp, the MOP Locus and Lr, the Requirements Locus. S(Lp) is the measure of the total MOP Locus.

The meaning of the term "measure" of the intersection or locus depends on the dimensionality of the parameter and performance spaces and whether the MOPs are continuous or discrete. For the class of systems where all MOPs are continuous, several cases can be considered. In the simplest case, there could be one Parameter, one MOP, and one Requirement. The MOP Locus would be a one dimensional line with is measure being defined as the length of a line segment. The Requirement would be represented as a point or set of points or segments of the same line. Furthermore, if there is one Parameter and any number of MOPs, then the MOP Locus is a line segment in the Performance Space. If there are two Parameters and two or more MOPs, then the MOP Locus is a surface in the Performance Space, and the measure becomes the area of the surface. If there are three parameters and three or more MOPs, then the MOP Locus is a volume in the performance space and the measure of the locus refers to the volume of the locus. At higher dimensionality of Parameter Space and Performance Space, measure refers to the hyper-volume of the locus. If the function representing one or more of MOPs is discrete, then the meaning of the measure must be modified. In these cases, only certain points, line segments, or areas can realized in the MOP space, and the calculation of the measure must be adjusted accordingly.

4.4.3 Requirements for SEAT

This section defines the Requirements that have been established for SEAT to support the performance evaluation segment of CAESAR II. The purpose of SEAT will be to produce plots of the various loci that can be used in performance evaluation of architectures, both fixed and adaptive ones. It also should be capable of calculating MOEs for an architecture for a given Parameter Locus, a mapping to a MOP Locus, and a set of Requirements.

Performance Evaluation Process Driven Requirements

The analysis of the requirements for SEAT is based on the process used in performance evaluation. The analysis begins with a set of performance data that has been obtained from the analyses of the executable model of the architecture developed in CAESAR II. In general, the data is contained in a matrix with each row containing the performance of the system for a particular operating point of the architecture. Thus, each row consists of the specific values of all the Parameters for the particular operating point and the corresponding values of the MOPs defined for the system. Given this data, the analyst will take a step-by-step approach to the performance evaluation. In general, the analyst will want to be able to view the Parameter Locus of the architecture to "see" the domain of all Parameters. If there is just a single Parameter, the plot will be a set of points on a single line segment. An example is shown in Figure 4.17. If there are two independent parameters, then a two dimensional plot would show a grid of parameter points (Figure 4.18). If the parameters are continuous with the data set tabulating performance at sample operating points, then it may be useful to interconnect the points in the graph to give the illusion of a surface as shown in Figure 4.19. With three Parameters, a three dimensional plot would show a set of stacked boxes. Since it is not possible to create plots of greater than three dimensions, if there are more than three Parameters, it may be necessary to view Parameter Loci created from subsets of the Parameter data.

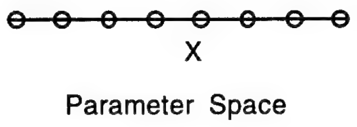


Figure 4.17 Example of a Parameter Locus for a Single Parameter

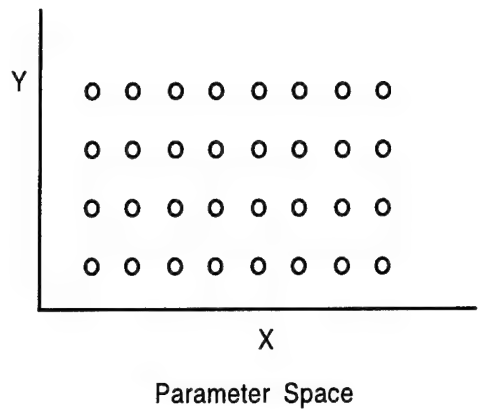


Figure 4.18. Example of a Parameter Locus for Two Independent Parameters

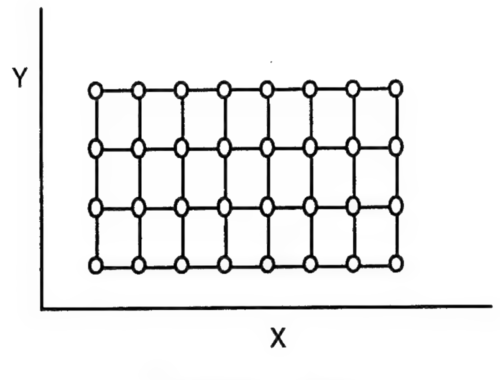


Figure 4.19 Two Dimension Parameter Locus with Interconnected Points

Next, the analyst will want to view the shape of the MOP Locus. The MOP Locus is the result of a mapping from the Parameter Locus to the MOP Locus. This means that each point in the Parameter Locus has a corresponding point on the MOP Locus. As was discussed in Section 4.4.2, the characteristics of the mapping depend on the number of Parameters and the number of MOPs. One parameter yields a line segment for the MOP Locus; two parameters yield a surface MOP Locus, etc. If the number of parameters exceeds the number of MOPs, then the MOP Locus will be a family of line segments, areas, or volumes. For example, if there are more than two Parameters, then the MOP Locus for two MOPs will be a family of surfaces. When there are three MOPs, the MOP Locus will be points in a line segment if there is a single Parameter, a surface if there are two Parameters, and a volume if there are three or more Parameters. If there are more than three MOPs, the analysts may wish to plot a set of sub-space MOP Loci, i.e., parametrize the MOP loci for fixed values of the remaining MOPs.

Figure 4.20 shows an example of the mapping from a simple two Parameter Locus to a two MOP Locus. If the Parameters and MOPs are continuous, it may be useful to connect the sample points to better illustrate the shape of the loci as shown in Figure 4.21. Such a plot is called a wire-mesh plot. The concept of a family of MOP surfaces is shown in Figure 4.22 where there are three parameters, X, Y, and Z. The loci over the range of X and Y for two different fixed values of Z are shown. The dotted connectors show how the plot would appear if the plotting routine drew a connector between all adjacent points in the Parameter Locus.

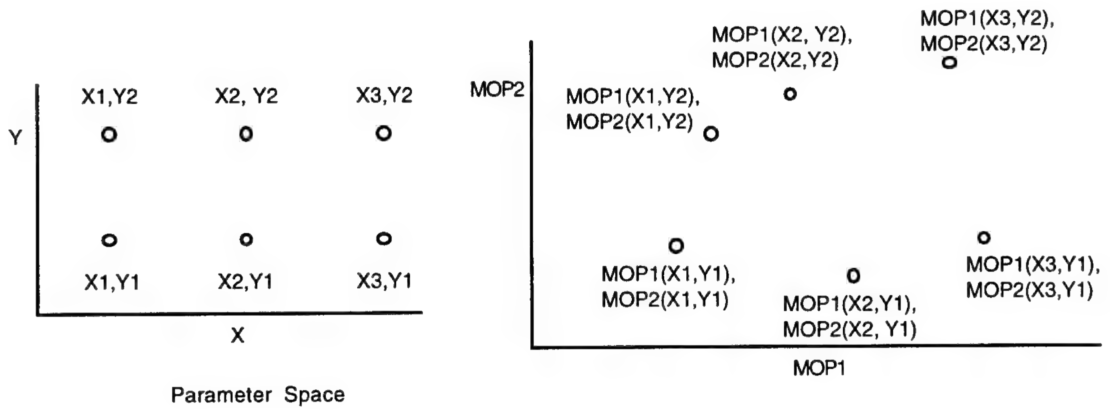


Figure 4.20 Mapping From Parameter to MOP Space.

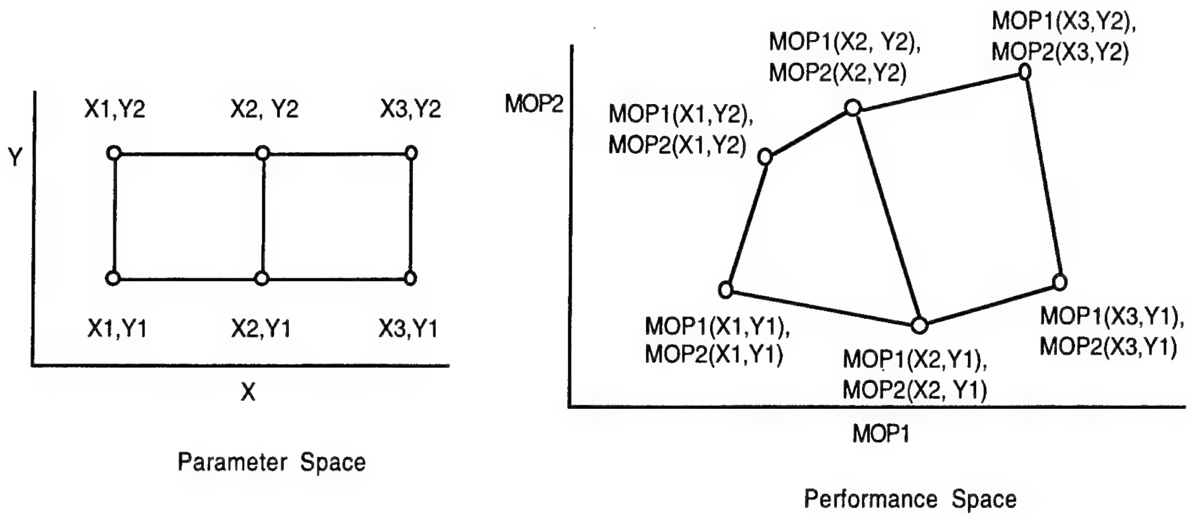
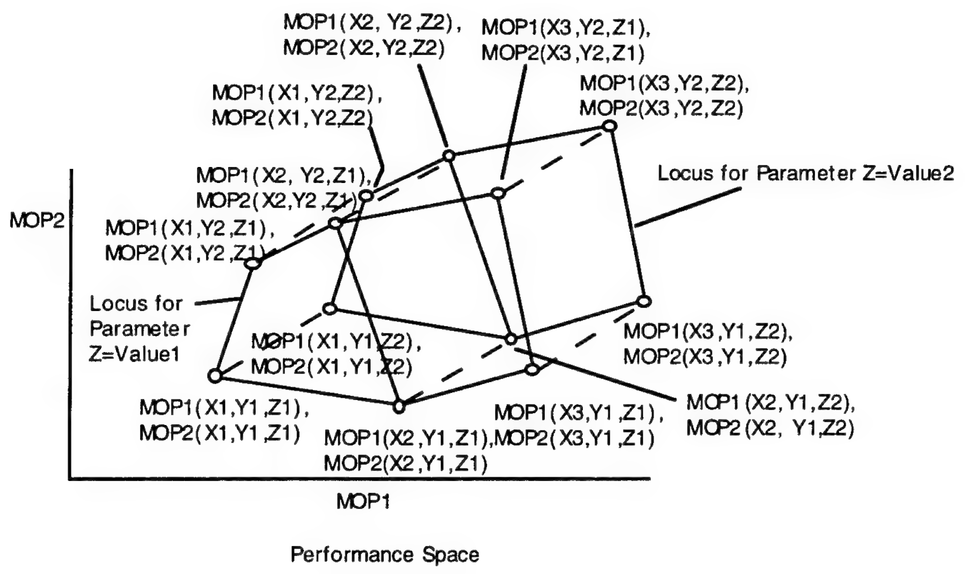


Figure 4.21 Mapping With Adjacent Points Connected



MOP1(X2, Y2,Z2), MOP2(X2,Y2,Z2)
MOP1(X3,Y2,Z1), MOP2(X3,Y2,Z1)
MOP1(X1,Y2,Z2), MOP2(X1,Y2,Z2)
MOP1(X3,Y2,Z2), MOP2(X3,Y2,Z2)
MOP1(X2, Y2,Z1), MOP2(X2,Y2,Z1)
Locus for Parameter Z=Value2
MOP2
MOP1(X1,Y2,Z1), MOP2(X1,Y2,Z1)
Locus for Parameter Z=Value1
MOP1(X1,Y1,Z2), MOP2(X1,Y1,Z2)
MOP1(X3,Y1,Z2), MOP2(X3,Y1,Z2)
MOP1(X1,Y1,Z1), MOP2(X1,Y1,Z1)
MOP1(X3,Y1,Z1), MOP2(X3,Y1,Z1)
MOP1 (X2,Y1,Z2), MOP2 (X2, Y1,Z2)
MOP1(X2,Y1,Z1), MOP2(X2,Y1,Z1)
MOP1
Performance Space

Figure 4.22 Family of MOP Loci

With three dimension MOP plots, the locus may be slightly distorted if the technique just described is not modified. This is caused by connecting four points in the three dimension performance space when there are two Parameters. Because three distinct points define a plane in three dimensions, there is no guarantee that the fourth point will be in the same plane as the other three. Thus the shaped plot creates an false illusion of a surface. A better method of generating the MOP plots would be to partition the parameter space into triangles and to create the surface plot from the mapping of these triangles in the three dimension performance space.

Each triangle would form a true plane in three dimensions. Figure 4.23 illustrates the concept. A similar concern occurs for the plots of three Parameters. Again the solution is to partition the parameters into tetrahedrons with triangular faces to ensure that the plot in three dimensions is accurate.

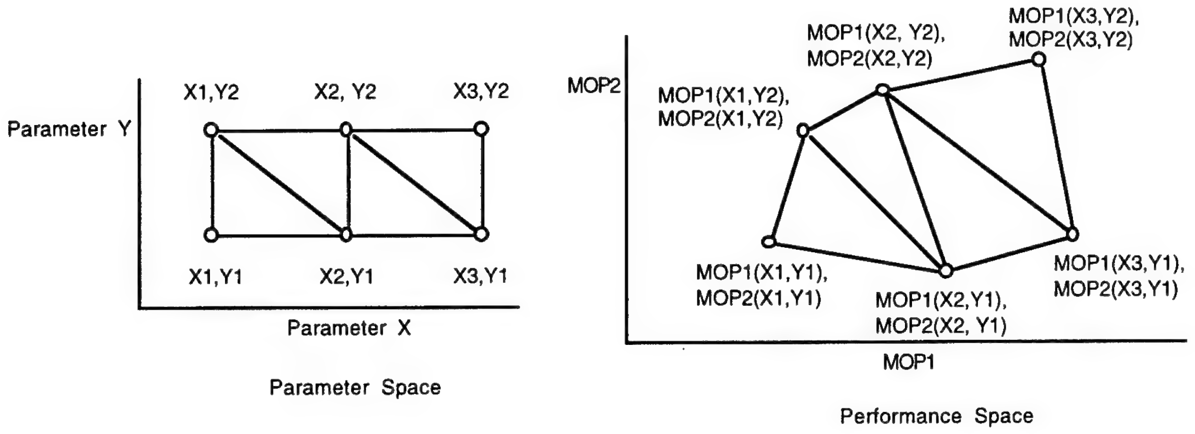


Figure 4.23 Three Point Wire-Mesh Plotting to Create Surfaces

Once the shapes of the MOP Locus is known, it may be worthwhile to view the value of specific MOPs versus the value of certain Parameters. Such plots can indicate whether the mapping from the Parameter Locus to the MOP Locus is a proper function or not. It can also indicate the sensitivity of changes in the values of the MOP to changes in the values of key Parameters. Such plots can be two dimensional showing the value of the MOP on the ordinate (y axis) and the value of the parameter on the abscissa. If there is more than one Parameter, the analyst will have to specify the values of the parameters that are not plotted or plot a family of plots that would be generated by systematically incrementing the values of the non-plotted Parameters and plotting the portion of the MOP Locus as the plotted Parameter is repeatedly varied over its domain. It may also be useful to create three dimension plots showing the value of a single MOP on one axis versus combinations of two parameters on the other two axis. Again, if the data set has more than two Parameters, then the analyst will either have to specify the value of the non-plotted parameters or plot a family of loci by incrementing through the values of the other parameters.

One of the main purposes of performance analysis is to determine if the architecture meets the Requirements of the system. The analyst may wish to plot the Requirements Locus in the same coordinate system as the MOPs. It should be noted that while many times the Requirements are independent and specified as a boundary value (e.g. the resolution shall be less than 3 meters; the workload may not exceed a certain level), this does not always have to be the case. Some Requirements may be specified as a function of more than one MOP. For example, a Requirement may by stated as a linear or non-linear combination of MOPs (e.g. $a*(MOPi) + b*(MOPj) < Requirement\_k$). Thus the Requirements Locus may be more complicated than a simple "box' in three dimension space.

Given the requirements, the analyst may want to plot an overlay of the Requirements Locus and the MOP Locus. Such plots can indicate how well an architecture meets requirements. It would be useful for the plot to visually indicate the portion of the MOP Locus that meets Requirements and the portion that does not. The plot should also show, by points or line segments, where the Requirements Locus and the MOP Locus intersect.

Once the MOPs and Requirements are known, the SEAT system should be capable of computing the MOE of the architecture and evaluating ways of improving the MOE. When a system does not fully meet the requirements, two options exist: change the design of the system so that it meets more or all of the Requirements or change the Requirements. One way to assist in the former option is to map from the MOP Requirements Locus back to the Parameter Locus to identify the partition of the Parameters Locus into the portion that cause the system to meet requirements and the portion that does not meet Requirements. Indeed, a MOE that differs from the one suggested in Equation 1 in Section 4.4.2 is the percentage of the Parameter Locus that causes the system to meet all the Requirements.

It should be pointed out that the reverse mapping may require some interpolation to determine the mapping of the intersection of the Requirements and MOP Locus to the Parameter Locus. This interpolation can be facilitated by calculating the points of intersection between the Requirements and MOP loci. Since these points are on the line segments that connect MOP values that are mapped from adjacent Parameter values in the Parameter Locus, they can be mapped backward from the MOP Locus to the Parameter Locus. Figure 4.24 illustrates a mapping of the intersection of Requirements and MOP Locus to the Parameter Locus. Once the reverse mapping has been accomplished, an MOE, "percentage of Parameter Locus that meets requirements" can be calculated by summing the individual "Measures" (area, volume, etc.) of the appropriate portion of the Parameter Locus and dividing by the total Parameter Locus measure. Figure 4.25 illustrates the concept where the area in the darkened border represents the portion of the Parameter Locus of Figure 4.24 that meets requirements. Similar principles could be illustrated for n dimension Performance Spaces and m dimension Parameter Spaces. In this case were $n = m = 3$, the calculation of the MOE will involve the summation (or integration) of a volume measure instead of an area measure. Algorithms to determine the intersections, partition the triangles (tetrahedrons), and sum the areas (volumes), will need to be developed.

Mapping from the MOP Locus to the Parameter Locus offers two advantages over performing MOE analysis in the Performance Space. The first is that it more clearly indicates the value range of the Parameters that cause the architecture to fail to meet requirements. This can be a first step to determining the components in the design of the architecture that directly affect the critical performance of the system. Increasing the capability of these components may lead to better performance in the areas of the MOP Locus that do not meet Requirements. The second advantage is computational. In many cases the Parameter Locus will consist of regular geometric shapes for which exact mensuration formulae exist. The corresponding MOP Locus will almost always consist of irregular shapes for which it is more difficult to compute the appropriate measure.

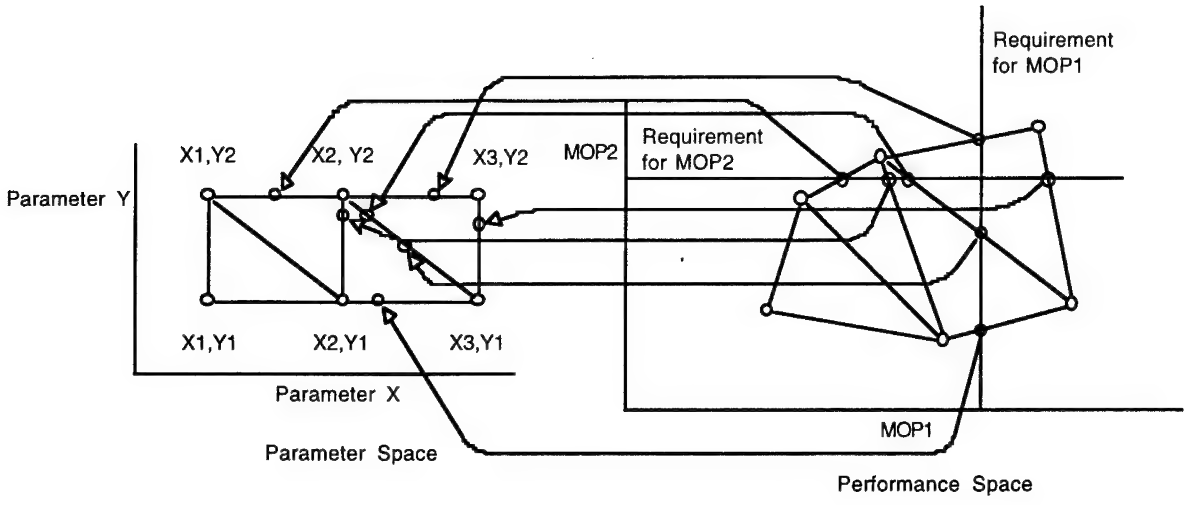


Figure 4.24 Mapping Requirements/MOP Loci Intersection Points to Parameter Locus

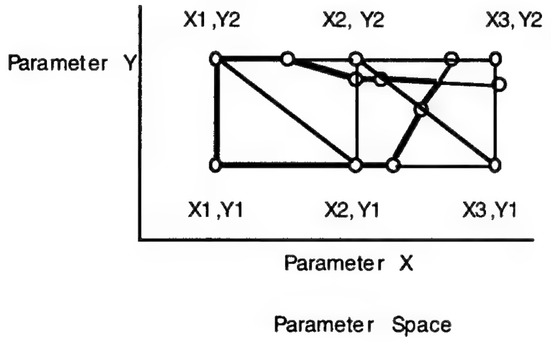


Figure 4.25 Determining the Portion of the Parameter Locus that Meets Requirements

When an architecture fails to meet all Requirements over its operating range and changing the design to meet Requirements is not desirable or even possible, a second option is to investigate changing the requirements. A capability to plot the value of an MOE versus the Requirements for MOPs can illustrate to which Requirements the MOE is most sensitive and indicate by how much a change in Requirements effects the MOE value. Two dimension plots can show the value of the MOE versus the changes in one Requirement, and three dimension plots can show the affect of changes in a combination of two Requirements. Recalling the formula for calculating an MOE, it should be noted that all MOPs and Requirements must be used in the calculation of the MOE even though only one or two Requirements are shown in the plot. This means that non-plotted Requirements must be set at some user specified value while the MOE is calculated as a function of changes in the plotted requirements. These plots can be obtained by sweeping or incrementing the value of the plotted Requirement(s) and computing the corresponding MOE until the maximum MOE value is reached. Figure 4.26 illustrates how the value of the MOE in Figures 4.24 and 4.25 would change as the requirement for one MOP (MOP2) was varied. In this plot, the Requirement for MOP1 was set at the maximum value of the MOP Locus. In the figure, when the Requirement for MOP2 is gradually relaxed (increased) from a value that precludes any of the operating points from meeting requirements, the area marked “a” would be the first part of the Parameter Locus that met requirements. As the requirement is further relaxed, the portion of the Parameter Locus meeting Requirements would grow until all of area “a”, then “b”, then “c” and finally all of the Parameter Locus met the requirements. The corresponding plot of the value of the MOE versus the value of the requirement for MOP2 is also shown in the figure.

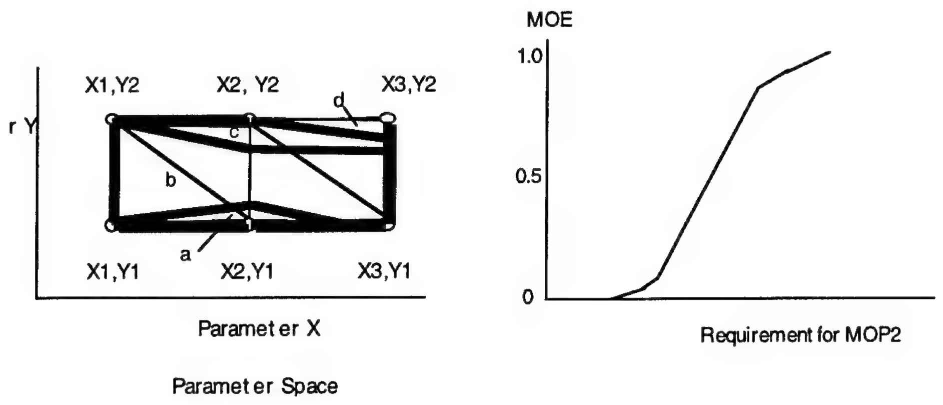


Figure 4.26 Calculation of the MOE as a Function of Change the Requirement for One MOP

There are other MOEs that could be useful in the evaluation of performance, and some can be more complex that the ones discussed to far. In some cases there may be a probability associated with the Parameter Locus. For example there may be a joint probability distribution associated with the Parameters. An MOE “the Probability that the architecture will meet requirements” could be computed by integrating the joint probability density function over portion of the Parameter Locus that meets requirements. In some applications, instead of a simple binary relation that MOPi meets or doesn’t meet a Requirement, there could be a range of values or utility associated with the value of each MOP. A joint utility distribution could exist for the Performance Space. A dual valued MOE “the minimum and maximum total utility value of the architecture” could be calculated by determining the smallest and the largest values of the combined utility for all MOP Requirements given the MOP Locus. This type of MOE could be appropriate if there was no knowledge about the likelihood of the actual operating point of the architecture. On the other hand, if a system has both a probability distribution over the Parameter Locus and a value or utility distribution over the Requirements Locus, the MOE “expected utility” could be computed. Finally, not all Requirements may have equal importance. A weighting system could be applied to degree each MOP meets requirements.

Summary of Requirements

Based on the description of the performance evaluation process, the following summarizes the functional requirements for SEAT.

R-0. Overall Requirement: To produce analyses and plots to support the performance evaluation of C3 architectures in the CAESAR II framework.

- R1. Receive, save, and print files and plots
  - R11. Receive and verify input file of performance data for any number of Parameters and MOPs
  - R12. Save calculations, data sets, and plots to user designated files
  - R13. Print selected files, data sets and sub-sets, and plots

R2. Calculate and plot user specified types of plots. Prompt user to provide needed inputs for plots.
   R21. Plot Parameter Locus for one, two and three parameters selected from the input by the user
   R22. Plot MOP Locus for one, two and three MOPs selected from the input by the user. User can fix values of selected Parameters or request "family of Loci"
   R23. Plot MOP values versus one or two Parameters. User specifies MOP and Parameters plus values of non-plotted Parameters.
   R24. Plot Requirements Locus for user specified Requirements or input file.
   R25. Plot Overlay of Requirements Locus and MOP Locus. Differentiate portions of MOP Locus that meet and do not met requirements. Indicate the locus of the intersection of the Requirements and MOP loci.
   R26. Plot the Parameter Locus with the mapping of the intersection of the Requirements and MOP Loci.
   R27. Plot MOE versus one or two Requirements. User is given the range of all MOPs and specifies the values of Requirements that are not plotted.

R3. Calculate user specified MOEs
   R31. Calculate Percentage of MOP Locus that meets requirements
   R32. Calculate Percentage of Parameter Locus that causes architecture to meet requirements
   R33. Calculate Probability that architecture will meet requirements. (User must provide joint probability distribution over the Parameter Locus.)
   R34. Calculate Minimum and Maximum Value or Utility of architecture. (User must provide value or utility functions for Requirements space. May include weighted utility)
   R35. Calculate Expected Utility
   R36. Other

R4. Allow user to specify ranges of axes, titles for axis and plots and type of plot (points, connected points, shaded areas or volumes.). Allow user to rotate plots to specific orientations or through a user specified range of orientations or perspectives.

R5. Generate user specified plots to the screen, printer, or file.

4.4.4 Design of SEAT

This section describes the preliminary design of SEAT. The design is presented as a $IDEF_0$ activity model that models the functionality of SEAT based on the functional requirements. The top two levels of the $IDEF_0$ model are presented. Figure 4.27 shows the context diagram of the design. It indicates the inputs, controls, and outputs of SEAT. Inputs include the input files that contain the parameter and performance data for the architecture, and the Requirements data for the MOPs. The controls include the System Effectiveness Analysis Procedures, which are not considered in the details of the $IDEF_0$ and the User Commands to SEAT. The outputs include Requests to the User for plotting, analysis information, and preferences, the Plots created by SEAT, and MOE values that the user may request. It is envisioned that the SEAT system will interact with the user through a graphical user interface. The Requests to User will appear in the form of windows with instructions, buttons, pull down menus, and text boxes. The User Commands will be provided through the GUI windows via the pull down menus, buttons, and text boxes.

The first level of decomposition for the design is shown in Figure 4.28. Notice the correspondence between the functions A1 through A5 and the Requirements summarized in the previous section. By following the description of the Performance Evaluation process in Section 4.4.3, it is possible to follow the key threads through the diagram. Each step of the process is handle by a function.

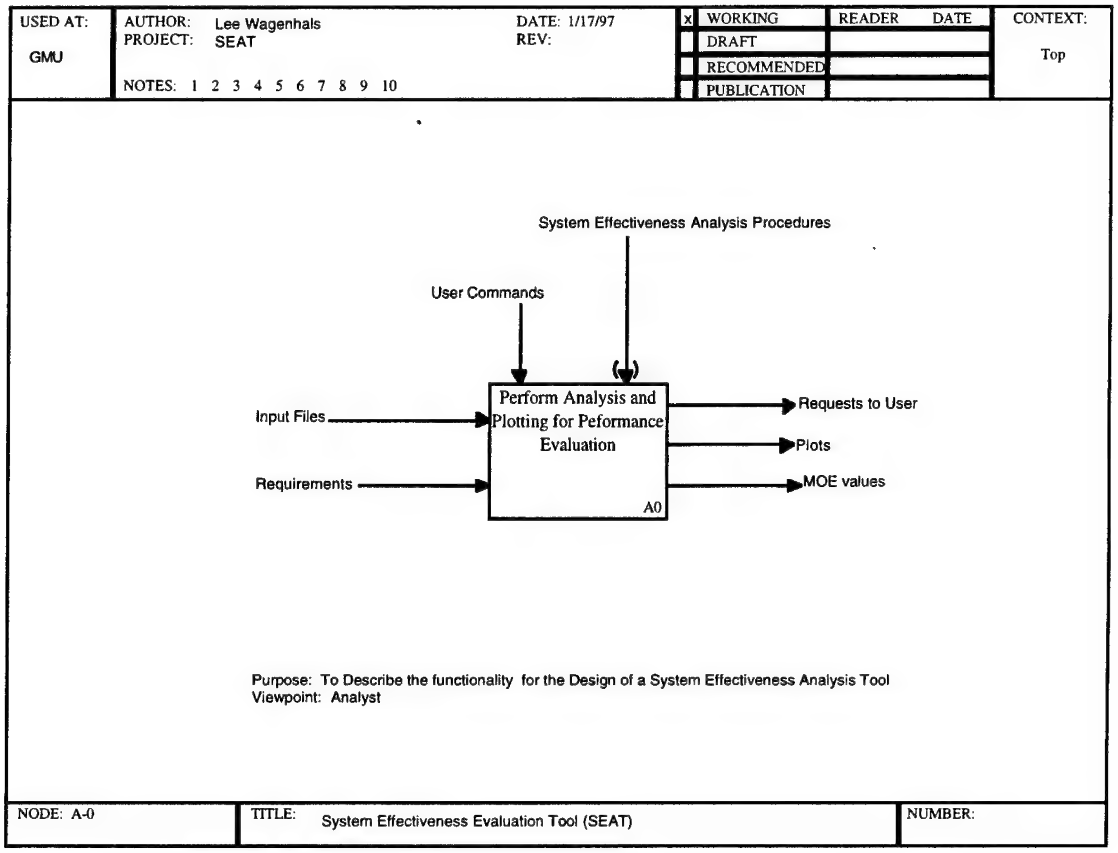

USED AT:
GMU
AUTHOR: Lee Wagenhals
PROJECT: SEAT
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 1/17/97
REV:
x WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
Top
System Effectiveness Analysis Procedures
User Commands
Input Files
Requirements
Perform Analysis and Plotting for Peformance Evaluation
A0
Requests to User
Plots
MOE values
Purpose: To Describe the functionality for the Design of a System Effectiveness Analysis Tool
Viewpoint: Analyst
NODE: A-0
TITLE: System Effectiveness Evaluation Tool (SEAT)
NUMBER:

Figure 4.27 Context Diagram of the SEAT Design

The first activity is reading and verifying the input file. The user will be given an window in which he provides the name and location of the input file. The activity reads and verifies the file. It can also print the file if requested. After the file is read, the second activity (A2) establishes, through a GUI-based dialog with the user, the type of analysis and plot to be accomplished. Options include Parameter, MOP, and Requirements Loci, MOP versus Parameter, overlay of Requirements and MOPs, and MOE versus Requirement plots. Requirements are input by the user for those plots that need them. The user can also request that SEAT calculate the MOE value to a given set of MOPs and Requirements. Such a request is transferred to the fourth activity, A4.

Once the type of plot and its parameters has be established, the third activity prepares the data for plotting. As part of that process, it has a dialog with the user to establish the plot preferences including plot and axis titles, axes ranges, type of mapping (point, wire-mesh, or shading), and the angle of orientation or rotation of the plot. If the plot involves the calculation of an MOE, the activity iteratively sends the MOE Parameters to A4 where the MOE is calculated. This would be the case for the MOE versus Requirements plots. Once the plotting data is prepared, it is sent to the fifth activity, A5, that generates the plot on the screen or saves it to a file. Once the plot is created, it can be modified by the user via the dialog performed by activity A3 where plotting preferences such as titles and viewing angles can be adjusted.

The fourth activity, A4, represents the MOE calculation operation. It contains the algorithms needed to compute the various types of MOEs that were described in Section 4.4.3. It includes the algorithms need to determine the intersection of the MOP and Requirements loci, perform the reverse mapping to the parameter space, and perform the numerical integration needed to calculate the percentage of the Parameter Locus that meets Requirements, the probability of meeting Requirements, the minimum and maximum utility, or the expected utility.

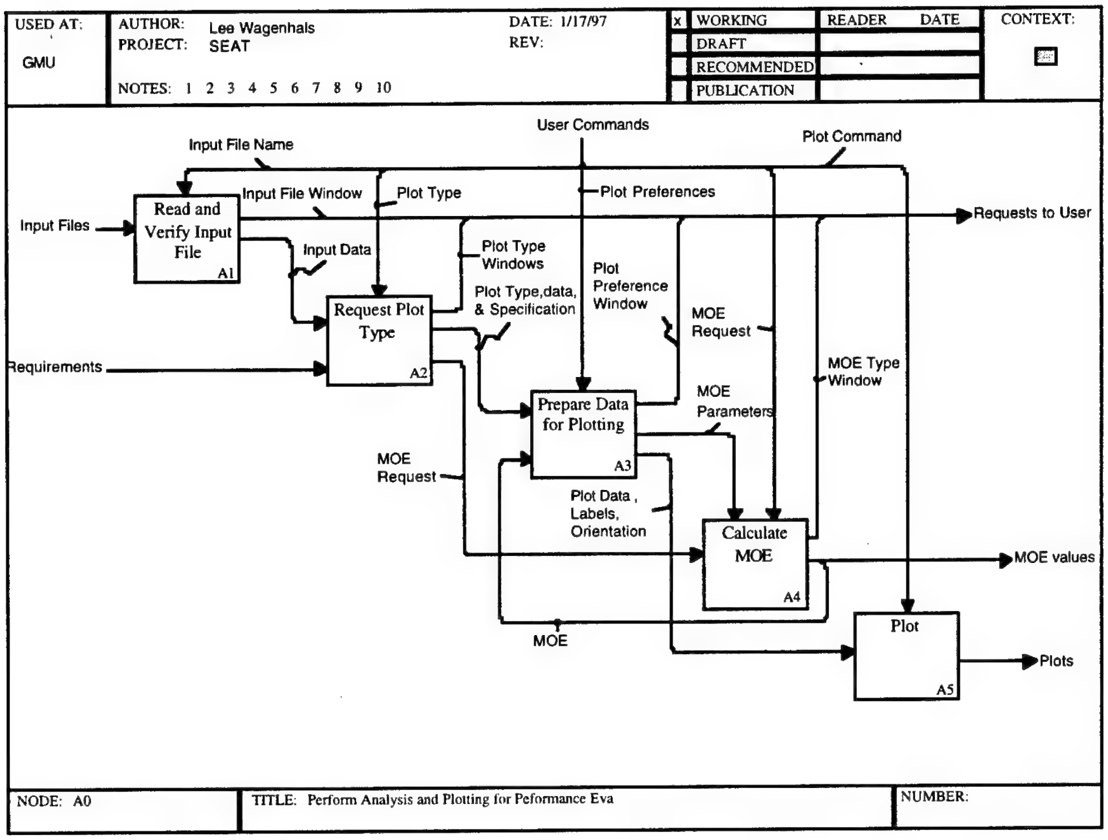


Figure 4.28. First Level of Decomposition or the Functional Design

4.4.5 Status of SEAT

Current Capability of SEAT

An experimental version of SEAT was developed to test the capability of MATLAB® to meet the requirements of SEAT and to begin the development and test of the many algorithms needed for the design. It consists of several proof-of principle modules that have been build using MATLAB® that provide partial functionality for each of the five functions shown in Figure 4.28. The experimental version of SEAT reads input files, interacts with the user to determine the type of plot to be created, allows the user to specify titles and rotation of the plots, computes an approximation of the MOE in Equation 1, and, of course, generates the plots on the screen. The user can copy and paste the plots into other documents using the operating environment of the computer.

Because the experimental version was for proof-of-principle, SEAT is currently is limited in the make-up of the input files and the types of plots it can handle. Furthermore, it is capable of calculating only one MOE for which it uses an approximation technique. Nevertheless, the experimental version of SEAT contributes to the effectiveness module of CAESAR II by allowing the user to produce two and three dimensional plots of the MOP Loci and a plot of an MOE versus MOP Requirements when given an input file containing Parameter and performance data. Three types of plots are available: a three dimension MOP Locus plot, a plot of MOPs versus one parameter, and plots of the MOE (defined by equation 1) as a function of changes in two Requirement variables.

The experimental version of SEAT can generate the MOP Locus in a three dimension performance space for any three out of three or more MOPs supplied by the input file. This plot is a surface for two Parameters and a volume for three parameters. For two parameter data, SEAT constructs the MOP plot by iteratively drawing small four sided polygons in the performance space, the vertices of which are the mapping from four adjacent values of Parameters from the Parameter Locus. The MOE versus MOP Requirements plot is a plot of the value of the MOE defined in Equation 1 as the Requirements associated with two user selected MOPs are varied in user specified increments over the range of these two Requirements. The values of the Requirements of the non-plotted MOPs are fixed at a value determined by the user. Examples plots that were copied and pasted from SEAT are shown in Figures 4.29, 4.30, and 4.31.

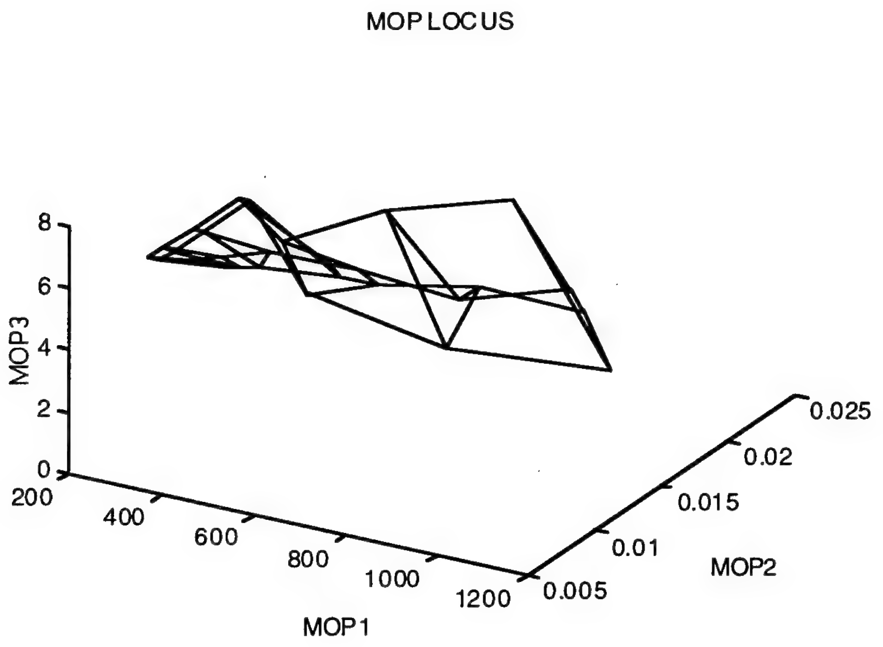


Figure 4.29 Sample MOP Locus Plot as a Wire Mesh

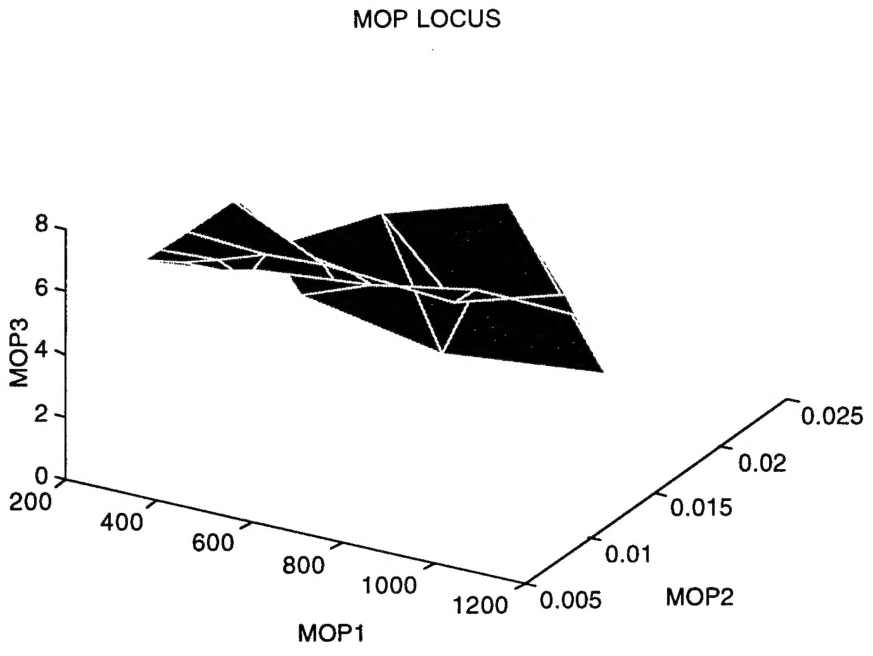


Figure 4.30 Sample MOP Locus Plot as a Surface

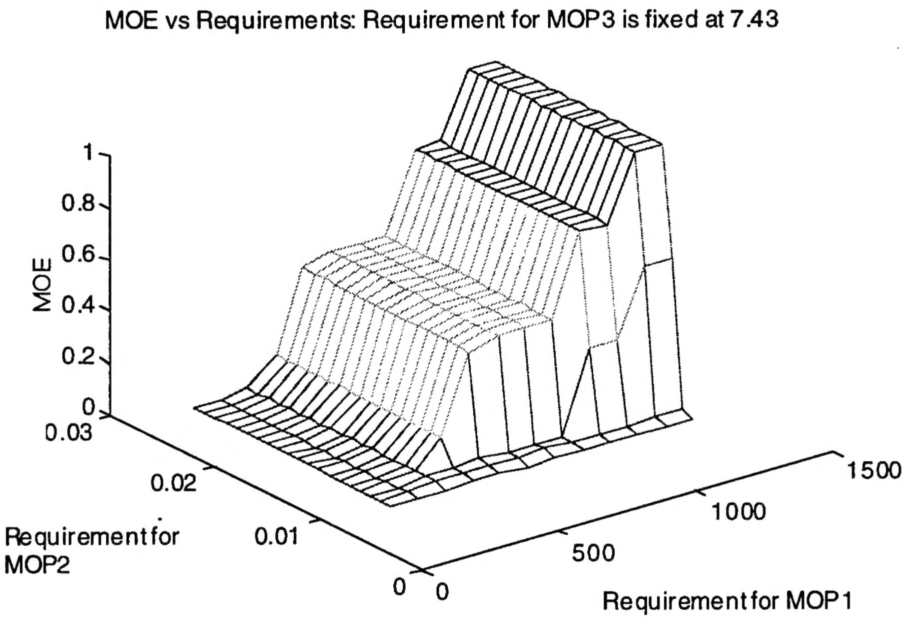


Figure 4.31 Sample MOE Versus Requirement Plot

SEAT creates its plots using an input file that contains a matrix relating Parameter points in Parameter Locus to points in the MOP Locus. While SEAT is being designed to work within the framework of CAESAR II, the input file can be generated by any technique that produces a mapping from a Parameter Locus to a MOP Locus. This could be accomplished using a pure analytical model, or in the CAESAR II framework, the user could have followed the first three stages and created an executable mode of the C3 architecture that is used to obtain the required mapping. The mapping is contained in a data file that is "read" by SEAT to produce the desired plots.

SEAT Future Improvements

The experimental version of SEAT was developed to test the capability of MATLAB® to meet the requirements of SEAT. With it, many of the concepts needed for the full design were tested including reading input files, generating windows for dialog with the user, calculating MOEs and generating wire-mesh and shaded plots. In the future, each of the modules will be expanded to increase the functionality to meet the entire design requirements.

4.5 SUMMARY

Progress achieved on all tasks during the period has been reported.

5. MEETINGS

Dr. Levis visited the the Naval Postgraduate School, Monterey, CA in August 1995 and briefed the staff on GMU research. Dr. Levis and Mr. Perdu attended the semi annual review of the A2C2 program at the Naval Postgraduate School, Montery, CA to brief the team on GMU research.

Drs. Levis and Buede and Messrs. Perdu and Wagenhals attended the 1996 Symposium on C2 Research and Technology at the Naval Postgraduate School in Monterey, CA and presented five papers derived from this research.

Dr. Levis and Mr. Perdu attended the TADMUS program review held at GMU.

Dr. Levis and Mr. Perdu attended the 2nd International Command and Control Research and Technology Sysmposium held at Market Bosworth, Warwickshire, England and presented two papers. Dr. Levis was a member of the International program Committee and chaired a session.

6. CHANGES

Changes in the scope of work of this project, as a result of two contract modifications and a contract renewal, have been documented in section 2.

7. CURRENT PERSONNEL

Dr. Buede completed his task and withdrew from the project at the end of the reporting period. Ms. Holly Handley, a PhD student, joined the project. Ms. Dinka graduated and obtained employment in industry. Mr. Tim Taylor and Mr. Alexey Yankovski contribute to the team.

Prof. Alexander H. Levis	Principal Investigator
Prof. Dennis Buede	
Dr. Abbas K. Zaidi	Consultant
Mr. Lee Wagenhals	Research Instructor (Ph.D.)
Mr. Didier Perdu	Research Instructor (Ph.D.)
Ms. Holly H. Handley	Graduate Res. Asst. (Ph.D.)
Mr. Eric Tsibertzopoulos	Graduate Research Asst. (MS)
[Ms. Etsiwohot Dinka	Undergraduate Research Assistant]
Mr. Alexey Yankovski	Undergraduate Research Assistant
Mr. Tim Taylor	Undergraduate Research Assistant

8. DOCUMENTATION

1. Hedy L. Rashba (1993). Problems in Concurrency and Coordination in Decision Making Organizations. Report GMU/C3I-143-R, C3I Center, George Mason University, Fairfax, VA.

2. A. H. Levis (1993). Adaptive Decision Making and Coordination in Variable Structure Organizations. Report GMU/C3I-147-IR, C3I Center, George Mason University, Fairfax, VA.

3. Zhenyi Jin (1994). Deadlock and Trap Analysis in Petri Nets. MS Thesis, Systems Engineering Department, George Mason University, Fairfax, VA.

4. A. H. Levis and D. M. Perdu (1994). Object Oriented Design of Decision Making Organizations, *Proc. 1994 The First Workshop on Command Informations Systems,* Oxon, UK. pp. 372-384. Defence Research Agency. Malvern.

5. A. K. Zaidi (1994). Validation and Verification of Decision Making Rules, Report GMU/C3I-155-TH, C3I Center, George Mason University, Fairfax, VA.

6. A. H. Levis and D. M. Perdu (1994). Object Oriented Design of Decision Making Organizations, A. H. Levis & I. S. Levis (Editors), *The Science of Command and Control: Part III, Coping with Change,* AFCEA International Press, Fairfax, VA.

7. A. H. Levis, D. M. Perdu and A. K. Zaidi (1994). Adaptive Decision Making and Coordination in Variable Structure Organizations. Report GMU/C3I-151-IR, C3I Center, George Mason University, Fairfax, VA.

8. A. K. Zaidi and A. H. Levis (1995). Rule Decomposition and Validation for Distributed Decision Making, *Proc. 1995 First International Symposium on Command and Control Research and Technology,* National Defense University, Washington, DC. pp. 210-217.

9. A. K. Zaidi and A. H. Levis (1995). Validation and Verification of Decision Making Rules, *Proc. 1995 6th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design and Evaluation of Man-Machine Systems,* MIT, Cambridge, MA. pp. 53-60.

10. A. H. Levis, D. M. Perdu & A. K. Zaidi (1995). CAESAR II: Computer Aided Evaluation Of System Architectures, Report GMU/C3I-162-R, C3I Center, George Mason University, Fairfax, VA.

11. A. K. Zaidi and A. H. Levis (1995). Object Oriented Design of a Multilevel Hierarchical Organization Structure, *Proc. US/Portugal Workshop On Underwater Vehicles and Intelligent Control,* Lisbon, Portugal. University of So. Louisianna Press.

12. A. K. Zaidi and A. H. Levis (1997).Validation and Verification of Decision Making Rules. *Automatica*, Vol. 33, No. 2, pp. 155 - 169, February 1997.

13. A. K. Zaidi and A. H. Levis (1996). On Generating DIS Architectures Using Genetic Algorithms, Paper GMU/C3I-166-P, C3I Center, George Mason University, Fairfax, VA.

14. Didier M. Perdu and Alexander H. Levis (1996). Object Oriented Design of An Air Combat Operations System Using Eagle Vision, *Proc. of the 1996 C2 Symposium,* Monterey, California, June 24 - 28.

15. Didier M. Perdu and Alexander H. Levis (1996). Distributed Process Coordination in Adaptive Command and Control Teams, *Proc. of the 1996 C2 Symposium,* Monterey, California, June 24 - 28.

16. Dennis M. Buede and Lee W. Wagenhals (1996). Influence Diagram Representation of Dynamic, Distributed Decision Making, *Proc. of the 1996 C2 Research and Technology Symposium,* Monterey, California, June 24 - 28.

17. Lee W. Wagenhals (1996). Digitization of the Battlefield: Approaches for Assessing Behavior and Performance of Distributed Command and Control Systems, *Proc. of the 1996 C2 Research and Technology Symposium,* Monterey, California, June 24 - 28.

18. Didier Leroy and Didier Hoffman (1996). Criteria of C3I Effectiveness, *Proc. of the 1996 C2 Research and Technology Symposium,* Monterey, California, June 24 - 28.

19. Levis, A. H. (1996). Adaptive Decision Making and Coordination in Variable Structure Organizations, Report GMU/C3I-176-IR, C3I Center, George Mason University, , Fairfax, VA.

20. Levis, A. H., D. M. Perdu, E. Tsibertzopoulos and A. Farshadfar (1996). A2C2 - The First Experiment: Architectural Considerations, Report GMU/C3I-177-R, C3I Center, George Mason University, Fairfax, VA.

21. Levis, A. H. & D.M. Perdu (1996). CAESAR II: A System for the Design and Evaluation of Command and Control Organizations, *Proceedings of the 1996 ICCRTS Conference,* Market Bosworth, England, September 23 - 25.

22. Levis, A. H. (1996). System Architectures. *Handbook on Systems Engineering and Management,* A.P. Sage and W.B. Rouse, Eds., Wiley, NY. (to appear in 1997)